Robin Bradbeer

LEARNING TO USE THE ZX SPECTRUM COMPUTER

Robin Bradbeer

LEARNING TO USE THE ZX SPECTRUM COMPUTER

This beginners' guide really does begin at the beginning. It assumes that you want to learn to *use* the ZX Spectrum in your work or leisure, not become a theorist in computing. *Learning to Use the ZX Spectrum* provides a simple, down-to-earth, jargon-free introduction to the machine and its software. Follow the text and illustrations and you will end up operating the ZX Spectrum and understanding its many capabilities.

Many applications of the ZX Spectrum are described, including business, educational and hobby uses. Additionally, a simple and direct introduction to programming the ZX Spectrum is given in a way which will help motivate the user to further investigation of the ZX Spectrum's capabilities. The ZX Spectrum's ability to produce and draw pictures and diagrams is explored and explained, and programs for a large number of graphics applications are presented.

This book will appeal to new ZX Spectrum owners, students in schools and colleges where ZX Spectrums are used, businessmen who wish to learn about how to use the ZX Spectrum and program it. It will help those who are already learning to use the ZX Spectrum, but find their current manuals difficult to follow. It also provides the 'would-be' purchaser of microcomputers with information on how the ZX Spectrum operates and performs, which will help him to assess whether the machine will suit his need.

About the series

This series of books has been designed to provide potential users, established users, teachers, students and businessmen with standardised introductions to the use of popular microcomputers. Extensive use has been made of photographs, diagrams and drawings to illustrate the text and make it easy to read and understand.

As the layout and content of the books in the series are similar, each book may be used in conjunction with others for purposes of comparison of performance and capabilities. The *Learning to Use* series is an inexpensive way of checking that the would-be purchasers' provisional choice of machine is the correct one.

The series is open-ended and will cover new models of microcomputers as they appear on the market.

Other titles in the series

Learning to Use the PET Computer
Learning to Use the BBC Microcomputer
Learning to Use the VIC-20 Computer
Learning to Use the ZX81 Computer

ISBN 0 566 03481 6
Gower Publishing Company Limited, Gower House, Croft Road,
Aldershot, Hampshire GU11 3HR, England

Robin Bradbeer LEARNING TO USE THE ZX SPECTRUM COMPUTER

Learning to Use the ZX Spectrum

Learning to Use the ZX Spectrum

by Robin Bradbeer

Gower

Published by
Gower Publishing Company Limited,
Gower House, Croft Road, Aldershot,
Hampshire GU11 3HR, England

British Library Cataloguing in Publication Data

Bradbeer, Robin
Learning to Use the ZX Spectrum
1. Sinclair ZX Spectrum (Computer)
I. Title
001.64'04 QA76.8.S62/

ISBN 0-566-03481-6

Ginn School Edition: ISBN 0-602-22633-3

Printed and bound in Great Britain
at The Pitman Press, Bath

Contents

List of Figures

Foreword

The ZX Spectrum, from Sinclair Research, is the latest in a very short line of microcomputers that started with the ZX80 in 1980 and then the ZX81 in 1981. Over one million of these computers have been sold.

The ZX Spectrum follows the design philosophy of these two previous computers but has the added bonus of colour, sound and much more memory. This book is designed to complement the instruction books provided by the manufacturers and takes some of the ideas just hinted at in these and shows the reader how to get more from their machine.

Like the other books in this series this book is not designed to be a programming manual. There are many others on the market that do this already. What it does do is introduce the reader to the computer in a rather painless way and hopefully an enjoyable way too. The introductory chapters take the reader carefully through setting up procedures and then on to very elementary programming. The later chapters investigate the graphics sound and colour capabilities of the system and are amply illustrated by a host of programs, all explained. Finally a series of appendices cover such topics as software, plug in attachments, magazines and books and compatability with preceding Sinclair computers.

I should like to thank Garry Marshall for doing the initial work on the first book in this series, and all those at Gower who patiently read, and reread, the manuscripts.

R.T.B.

Chapter 1
Introduction to the ZX Spectrum

What is the ZX Spectrum?

The ZX Spectrum is a computer. It is usually called a microcomputer because it is extremely small compared to early computers, and also because its electronic 'heart' is a microprocessor. As you can see from Figure 1.1 overleaf, the ZX Spectrum has a keyboard, which is set out in the same way as that of a conventional typewriter, although using a smaller keyboard than that of a normal typewriter. Inside, as is shown in Figure 1.2 overleaf, there are a number of integrated circuits, one of which is the microprocessor, while others provide the ZX Spectrum memory and can store information. Initially, there is no need to worry about what is inside the ZX Spectrum. The electronic circuitry and devices which make the ZX Spectrum work are fascinating, but a detailed understanding of them is not necessary in order to make use of the ZX Spectrum – and this introductory book is about using the ZX Spectrum.

The keyboard is for communicating with the ZX Spectrum: commands for it to obey and information for it to store can simply be typed out. Because the keyboard is arranged in the same way as that of a typewriter, those with a knowledge of typing will be able to find their way round easily. Anything typed on the keyboard is automatically displayed on the television screen.

The ZX Spectrum possesses a number of screen editing facilities, so that typing errors can be simply corrected with the aid of special keys, and amendments can be made in a simple and natural fashion. In fact, the ZX Spectrum's screen editing system has been very carefully thought out, and is easy to use after a little practice. It works in the natural way that most people expect it to.

Besides letters and numbers, the ZX Spectrum can also produce graphics symbols with the aid of which it is possible to type out a picture in much the same way as a paragraph of text is typed. Figure 1.3 shows a drawing that has been produced by the ZX Spectrum on a television screen. This easy access to graphics is a tremendous bonus, for the imaginative use of graphics enlivens all sorts of applications, ranging from games programs to educational programs and the presentation of

Figure 1.1 The ZX Spectrum

Figure 1.2 Inside the ZX Spectrum.

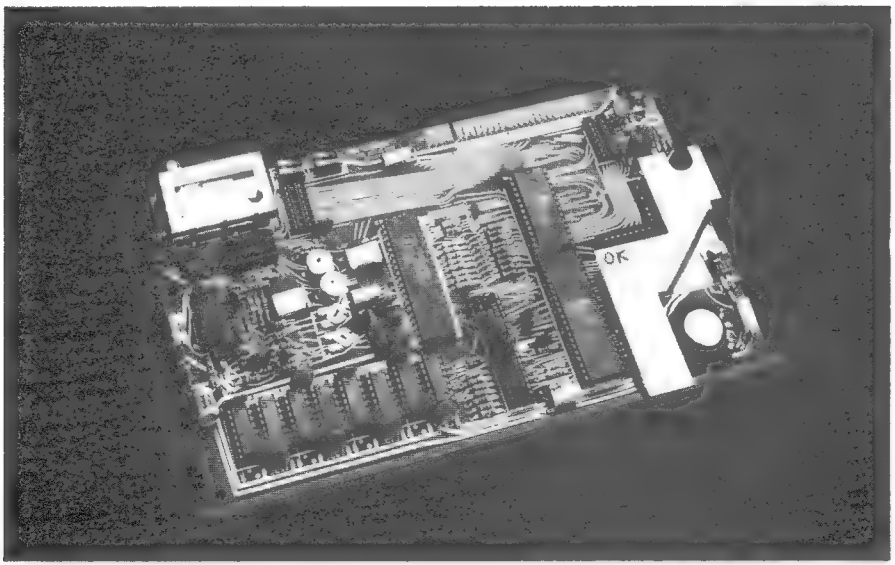
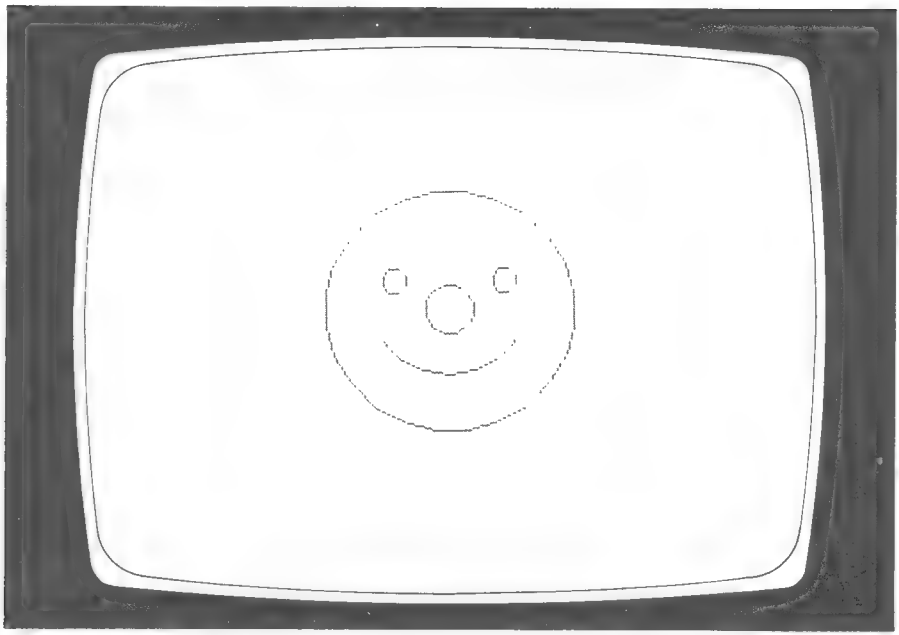


Figure 1.3 A drawing on the ZX Spectrum screen.



information in a business package.

So the ZX Spectrum is a computer that is small enough to be easily portable and to use in the home. To use it, all that it is necessary is to plug the adaptor into the mains electricity supply, plug a television into the appropriate socket on the side and turn it on. Further, as soon as it is switched on, a computer language with which to communicate with the ZX Spectrum automatically becomes available. This language is BASIC, and it enables the user to issue commands which are promptly and automatically obeyed by the ZX Spectrum.

How was it developed?

The event that signalled the advent of microelectronics, and hence of microcomputers, was the space race of the 1960s between the USA and Russia. Because American rockets were less powerful than those of the Russians, the Americans needed to reduce the weight, and hence the size, of everything that had to be carried by their rockets including all the electronics. This stimulated the American electronics industry to investigate and develop means of miniaturising their electronic circuitry. These developments culminated in the microprocessor, which in addition

to being extremely small (given its capabilities) is a multi-purpose device in the sense that it is programmable, and can perform any electronic function for which it can be programmed. This versatility has led to the use of microprocessors in a tremendous number of applications and the consequent mass production of microprocessors has caused the cost per unit to drop to a matter of a few pence.

The ZX Spectrum is based on the Z80 microprocessor, which is manufactured by NEC of Japan. In a sense, a ZX Spectrum is an easy-to-use microprocessor because although the microprocessor in the ZX Spectrum actually does all the hard work – all the computing – it has been made the core of a system designed in such a way that the user can tap its potential without necessarily needing any knowledge of the details of how it works.

Incidentally, the microprocessor used in the ZX Spectrum is the same as that used in many applications in industry and commerce. It is probably the most popular microprocessor in use today. Ironically, given the military origins of microelectronics, these microprocessors are more advanced examples of the technology than those used in the guidance systems of inter-continental ballistic missiles!

The ZX Spectrum is the third in series of small computers made by Sinclair Research, based in Cambridge, England. The first was the ZX80, introduced in 1980, and this was the first computer to sell for less than £100. Its successor, the ZX81, was introduced in 1981, and was essentially an upgrade of the ZX80. The total worldwide sales of these two machines is expected to reach one million by the beginning of 1983. This makes them the world's best selling microcomputer. Both were aimed at the domestic market and were designed to fit on to a domestic TV set. They were very limited in their capabilities but were ideally designed for the first-time user who did not want to spend too much money getting started. The Spectrum is altogether a different machine, with colour graphics, sound and the ability to have many different peripherals attached to it. The Spectrum comes in two models, one with 16K bytes of memory, the other with 48K bytes. The Spectrum is not totally compatible with the ZX81/80 and a list of the more obvious differences is given in Appendix 2.

An important difference between the different models is that they make available different amounts of memory to the user, so that, for example, the 48K model provides treble the storage capacity of the 16K model. Any program that is to be used on the ZX Spectrum requires a certain amount of memory in which to store it, and the more memory that your Spectrum possesses the wider will be the range of programs that it can store and run. Programs of any length, including many business programs, can require a great deal of memory. As a rough guide, the 48K

Spectrum is almost certainly necessary for sophisticated business packages, while for running games programs the 16K Spectrum may well be adequate.

What can it do?

Fundamentally, a ZX Spectrum can do anything that you can tell it to do, which is to say that it will obey any instruction or set of instructions that it is given. A set of instructions to the computer is usually called a computer program, so the ZX Spectrum, like any other computer, executes programs and does what they tell it to do. Thus one way to make use of a ZX Spectrum is to learn to program it in its own language. As we have seen, its natural language is BASIC. This is a simple programming language that is designed to be easy both to learn and to use. It is not understood directly by the microprocessor in the ZX Spectrum, and is translated to the rather restricted code understood by the microprocessor which is known as its machine code. It is also possible to program the ZX Spectrum in machine code, and although this is considerably more difficult to do, it gives programs that are executed more quickly because the translation stage is omitted.

However, it is not essential to be an expert programmer to use the ZX Spectrum, since programs can be obtained for it from a number of commercial sources. These programs are recorded on cassettes. The cassettes are of the same type as good quality audio cassettes: Figure 1.4 shows a cassette and a microdrive. To transfer a program from a cassette to

Figure 1.4 A cassette and a microdrive.

the ZX Spectrum so that it can run the program a cassette recorder must be used. Figure 1.5 shows a cassette unit attached to the ZX Spectrum. Programs can be obtained from Sinclair which maintains a small catalogue, including games and educational programs. There are many other companies which sell programs for the ZX Spectrum and which advertise regularly in the popular computing magazines.

These examples illustrate that the ZX Spectrum can be used in a variety of ways without the user having any knowledge of how to program it. Nevertheless, it is often useful to be able to program, if only to amend or modify an existing program. Besides, programming is fun, it is easy to do, and it provides a means of expressing and communicating your own ideas to the computer so that it can test them for you.

How can the ZX Spectrum be extended?

Besides performing computations and storing information the ZX Spectrum can, again like any other computer, be used to control other devices. Units that can be connected to the ZX Spectrum and whose operation can then be controlled by it are called 'peripherals'. We have

Figure 1.5 A cassette unit attached to the ZX Spectrum.

already met one of them in the cassette recorder. These units are the ones that are used for permanent storage of information by means of magnetic patterns on tape.

It is useful in many applications to have printed output, for example to provide a permanent record of the results of a computation, and for this purpose the ZX printer can be attached to the ZX Spectrum (see Figure 1.6). The ZX printers are able to reproduce the graphics symbols as well as

Figure 1.6 A printer attached to the ZX Spectrum.

letters and numbers so that it can print pictures as well as text.

Other than the tape recorder, power supply and television set, other peripherals can be attached to the ZX Spectrum. We have already seen that the ZX printer will give a printed output. Sinclair also make a number of other devices that can be added to the computer. These include fast and large memory devices like the Microfloppy and interface units that allow the Spectrum to be connected to other computers and to normal communications networks and standard printers. Other manufacturers also make peripherals for the ZX range of computers, and although the connections and driving software are different for the Spectrum compared to the ZX81 most of the latter's peripherals can be adapted quite easily.

Any peripherals attached to the ZX Spectrum use the connectors at the rear. Figure 1.7 shows the back view of the ZX Spectrum with the connection sockets for peripherals. At the rear of the computer are four jack sockets. From left to right, these are for power supply (9 volts dc from mains adaptor), two tape recorder connections and a video output that connects to the television set. Page 6 of the Sinclair Introductory booklet explains these connections in detail. Also at the rear of the computer is an edge connector that allows peripherals to be connected.

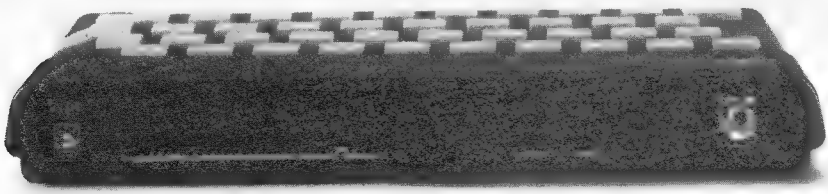
What are some typical applications of the ZX Spectrum?

The ZX Spectrum and its predecessors, the ZX80 and ZX81, have been used in many applications since their introduction in 1980. The areas in which they have been used can broadly be classified as in the home for recreational and personal use, in educational institutions, and in industry.

For personal use there are games programs of many kinds. Using the ZX Spectrum for playing games has frequently been criticised as a frivolous use of a personal computer, and there is no doubt that many games are lighthearted, but even these can provide relaxation and entertainment. However, there are many imaginative games which also have educational value. They can teach or help to develop attributes ranging from simple manipulation and coordination skills in children to the mental disciplines required to find solution methods for puzzles and to test strategy and tactics in games played against the computer. There are now chess-playing programs of a formidable standard, and it would seem perverse to argue that playing chess is a frivolous activity.

The presence of a ZX Spectrum in the home means that educational activities need not be restricted to schools and colleges. Simple computer-assisted learning packages are available for the ZX Spectrum, and they can be used just as effectively at home as at school. Computer-assisted learning is not, in any case, intended to replace teachers, but to assist them by

Figure 1.7 Back view of the ZX Spectrum, showing connectors.



providing another educational tool. In a post-industrial society it is important to expose everyone, and particularly children, to the current technology so that they may be aware of it, appreciate its capabilities, and thus be in a position to take advantage of it or to develop it in some way. The presence of a ZX Spectrum as an everyday item in the home or at school can help to achieve this objective.

In schools, the ZX Spectrum has many valid uses ranging from computer-assisted learning to the use of quiz programs to instructional programs. There are excellent examples of the latter dealing with how to program in BASIC which have been used in many schools, colleges and universities.

The colour and sound capabilities of the Spectrum make it ideal for educational work, especially at a primary school level. The ability to have quite high resolution graphics and the ease of use make it very good in the classroom. The large memory storage of the Spectrum also means that it can be used in the small business area, although it is not really designed for this purpose. The keyboard mitigates against it being used as a word processing system, but the addition of a decent printer and the Microfloppy drives make it very useful as a means of storing and manipulating large amounts of data. In any activity where large amounts of information have to be stored and certain items have to be retrieved, it is almost essential to use the Microfloppy rather than a cassette unit for storage. This is not only because a Microfloppy has a greater storage capacity, but also because it permits an item of information to be accessed more rapidly. Any item stored on a Microfloppy can be accessed almost at once regardless of its position. By contrast, on a cassette it is necessary to wind the tape as far as the required item, so that to reach some items may take rather a long time.

VisiCalc type programs have proved so successful as planning aids for managers that computers have been purchased purely to run more programs. They are planning and modelling programs that facilitate planning and forecasting by allowing the user to model his business and then to examine the effects on it that result from making particular changes. These programs are one example of how support for microcomputers has, in places, outstripped the facilities offered by the part of the computer industry that is geared to larger computers. Conventional computer people have been known to be contemptuous of microcomputers and their capabilities, but the enterprise shown in the microcomputer world has led to some rather abrupt reversals in attitudes.

Several programs for filing information are available for microcomputers. They are called, variously, 'data management systems', 'database systems' and 'information retrieval systems'. In essence they

permit the user to format, or structure, information, to store as much information as required and then to search the stored information for items having particular properties so that they can be displayed or typed out.

All these activities can be performed using existing programs. By learning to program the ZX Spectrum, you can write your own programs, not only to express your own ideas, but also because, as in most spheres, what you can buy seldom does exactly what you would like it to. Since the ZX Spectrum is much faster and more reliable than any person at activities such as numerical calculations, it would seem sensible to get the ZX Spectrum to do them. Besides acting as a source of entertainment, education and, in a limited way, as a business convenience, the ZX Spectrum can, in this information age, be used to extend and amplify the human brain.

Summary

The ZX Spectrum is a small computer, or microcomputer. The major novelty of a microcomputer is its size rather than its ability to compute. However, the smallness of a microcomputer means that its computing power is available so conveniently that it can be used as a personal tool in the home or at work. This small size is a direct result of recent technological developments.

The ZX Spectrum can be used in many ways, but its main areas of application are in business, education and for personal entertainment. Because programs can be bought to perform many tasks in these areas, the ZX Spectrum can be usefully employed almost as soon as it is acquired and with a minimum of expertise. The user can also run self-written programs using the BASIC language, which is not difficult to learn.

The capabilities of a ZX Spectrum can be extended in a variety of ways either by adding extras to the ZX Spectrum itself or by the acquisition of further units such as a printer, which can be attached to the ZX Spectrum and used in conjunction with it.

Chapter 2
Using the ZX Spectrum

Switching on

When the ZX Spectrum is connected to a television set and switched on a display such as the one shown in Figure 2.1 appears on the screen. The Sinclair copyright message appears in the bottom of the screen. This indicates that the computer is now ready for use.

The screen is wide enough to take 32 symbols, or characters, on one line. As soon as 32 characters have been placed on the same line, the cursor automatically moves to the beginning of the next line, and the next character to be typed will be placed there. In fact the screen can hold 22 lines, each of 32 characters, so that a symbol can be placed in any of $22 \times 32 = 704$ positions on the screen. (Two other lines are available at the bottom of the screen for messages.)

Figure 2.1 Screen display when the ZX Spectrum is switched on.

Loading a program

Probably the most enjoyable and painless way to become familiar with the ZX Spectrum and its keyboard is to use the ZX Spectrum to play a game that requires the user to give responses from the keyboard. Popular games such as 'space invaders' or 'sub-bomb' are ideal for this purpose.

Loading a program from a cassette into the Spectrum is very simple. The Sinclair Introductory Booklet gives a full account of how to do this so the following is a brief synopsis. First make sure that the cassette recorder is connected to the computer. For loading it is only necessary to have one lead connected, i.e. the lead from the external speaker or earphone socket on the recorder to the ear socket at the rear of the Spectrum. Insert the cassette containing the program and make sure that it is completely rewound, or if the recorder has a digital counter make sure that the tape is in the right place. It is a good idea to clear the computer screen before starting the procedure as characters on the screen can sometimes make it difficult to see what is happening. This can be achieved by keying the CLS command (on the V key), and keying ENTER. The series of letters and numbers will appear at the bottom of the screen, which tells us that everything is 'OK'. The simplest way of clearing the screen and anything that may be in the computer's memory is to turn the computer off and then on again! If you do this then the copyright message should appear as before.

Pressing the J key will cause the word LOAD to appear at the bottom of the screen. To load in the program we need to tell the computer what the name of the program is and we need to do this exactly. The exact name is typed in following a quote (") symbol which is obtained by pressing the P key at the same time as the red symbols shift key. When the exact name has been keyed in it is followed by another quote symbol.

The PLAY key of the recorder should now be pushed and followed by the ENTER key on the computer. The screen will clear and the border will change colour. If the television is tuned in correctly the following sequence should take place. Of course if you are using a black and white television set you will only see grey shades not the colours indicated!

Blue and red flashing border until the program is encountered.
Blue and red flashing stripes in the border area followed by the name of the program being loaded.
A burst of yellow and blue stripes as the program is being loaded.
A message at the bottom of the screen telling you that it has loaded 'OK'.

The ZX Spectrum has now loaded the program into the computer memory. To make the program work you now have to tell the computer to RUN it. This is done by keying RUN, which is on the R key. When

ENTER is pressed the program should work. If you have loaded a games program for example it should then instruct you how to play the game.

The complete dialogue produced by this loading procedure is shown in Figure 2.2. Leaving out the program name in the above procedure always causes the first program encountered on the tape to be copied into the ZX Spectrum memory. If there is only one program on the cassette ZX Spectrum, the name of the program can be omitted. Thus to load the first program from the cassette key in LOAD " " then ENTER. By using the program name in the loading command, a program that is not first on the cassette can be automatically loaded by issuing a single command. Bear in mind that if no program on the cassette bears the name you use, a long wait must be endured while the ZX Spectrum searches fruitlessly. Generally, it is sensible to have programs recorded on short C12 cassettes, and to have one program, but certainly not more than two, recorded on each side.

The keyboard

As we have seen, most of the keys have a number of functions attributed to them. Some of the keys produce special effects, such as clearing the screen, while many of the keys have drawing symbols on them. The special effect and drawing symbol keys are explained in this section.

Figure 2.2 Dialogue after loading the program "Wraptrap" from cassette.

Because the ZX Spectrum uses single keys to enter its BASIC commands the computer requires one of these at the beginning of each line. Therefore it is not possible to type directly to the screen immediately. These keywords are indicated by the inscription on each key. For example, if the letter P is depressed while the cursor displays the 'reverse K' then the instruction PRINT will appear.

As the keyboard mode is so important to understand it is useful to summarise what happens.

The flashing character is called the cursor. It shows you whereabouts on the screen the computer will put the next thing that you type. If you turn the computer off and on and then press ENTER, the copyright message will change into a K cursor. The letter that it uses tells you how the computer will interpret the next thing that you type. At the beginning of a line, it will be a flashing K, standing for 'keyword'. (The copyright message and reports also count as a flashing K.) A keyword is one of the computer's special words, occurring at the beginning of a command to give the computer a general idea of what the command is going to tell it to do. Since the computer is expecting a keyword at the beginning of a line, when you press – say – the P key, the computer decides not to interpret this as a P, but as PRINT; and it warns you that it is going to do this by making the cursor a K. When it has the first keyword, it does not expect any more of them, so what you type now will be interpreted as letters. To show this, the computer changes the cursor to an L – for 'letter'.

These different states are often called *modes* – e.g. keyword (or K) mode, and letter (or L) mode.

If you want to type a number of capital letters without holding capital shift down, you can make all letters come out as capitals by first pressing CAPS LOCK (capitals shift with the 2). To show this is happening, the L cursor will be replaced by a flashing C (for 'capitals'). To get lower case letters and the L cursor back, press CAPS LOCK a second time.

If you do this during keyword mode, you will not immediately notice any difference, but you will still see the effect later when the computer comes to choose between L and C.

As well as keywords, numbers and various programming and scientific expressions, the keyboard also has eight graphics characters. These appear on the number keys 1 to 8, and can be printed on to the screen in a similar way to letters and numbers. To do this the keyboard must be changed to graphics mode. This is done by pressing the capitals shift key with the 9. Notice the cursor change to a 'G'; pressing the 9 key will change back to 'L' mode.

There is one last mode that the keyboard can be changed to. The extended mode, indicated by a 'E' in the cursor, is obtained by pressing

capitals shift and symbols shift at the same time. This allows most of the scientific and programming functions to be used. Pressing the two shifts will revert the keyboard back to lower case letter, L, mode.

The keyboard itself is colour coded. To get the red words *on* the keys entered you have to use the symbols shift key *with* the key containing the word. We have already seen this with the " symbol on the P key. The green keywords on the metal above the keys are obtained using the E mode, whilst the red keywords below the keys are obtained by using the E mode with a shift key. For example to get the INT, press both shift keys and then the R key. To get the VERIFY keyword press both shift keys and then the R key with the CAPS shift key. Although this may seem strange at first a little practice will soon allow you to use the keyboard very easily.

The number keys work a little differently from the letter keys and their operation is discussed in Chapter 4 covering graphics.

CONTROL keys

We have already seen that there are a number of control keys on the Spectrum that do not actually cause characters to be written on the screen, although they do cause things to happen inside the computer. There are a number of others that do the same thing.

The BREAK and RUN keys

The BREAK key is used to stop a program that is being RUN by the Spectrum. The RUN key causes the program that is in the Spectrum's memory to be executed.

The DELETE key

Pressing the DELETE key causes the character to the left of the cursor to be deleted and moves the cursor one place to the left. This can be used with the edit keys.

The CLS key

This causes the screen to be cleared. It can be used in a program or directly. It moves the cursor to the bottom left hand of the screen.

The cursor control keys

These keys permit the cursor to be moved around the screen, when editing is performed, in indicated directions. The screen contents are not changed as the cursor is moved.

The GRAPHICS key

Holding down the CAPS SHIFT key and pressing the 9 key causes a G

cursor or graphics cursor to appear in the cursor position. The graphics symbols can then be used to compose pictures of surprising detail and complexity. Figure 2.3 shows a 'space invader' drawn on the ZX Spectrum screen together with the drawing symbols from which it is composed.

Editing

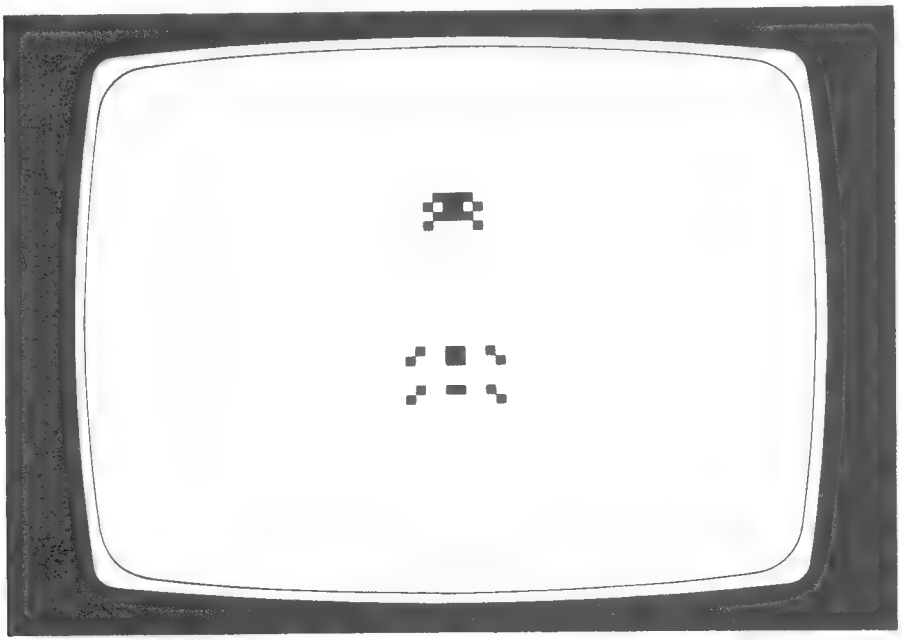
Editing is very simple, but can only be performed on a listed program. This feature is considered later, when simple programming is performed.

Giving simple commands to the ZX Spectrum

The ZX Spectrum can be used in a mode in which it obeys individual commands given to it at once. These commands have to be expressed in the BASIC language. A few commands of this kind, including LOAD and RUN, have already been introduced.

Generally speaking, what a computer does in most situations is to accept and store information of some kind, manipulate or process it, and then give the results in some appropriate form. Individual aspects of this can be illustrated by performing them with single commands. For

Figure 2.3 'Space Invader' and the graphics characters from which it is made.

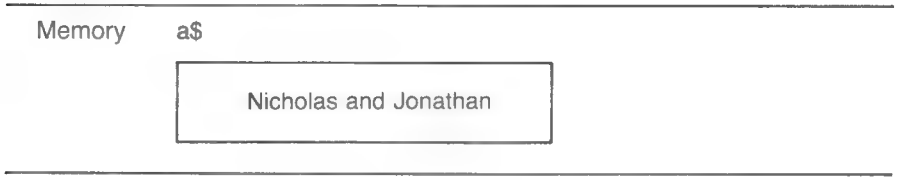


example, a string of characters can be stored in the ZX Spectrum's memory by typing on a new line

```
LET a$ = "Nicholas and Jonathan"
```

and pressing the ENTER key. Pressing ENTER causes the ZX Spectrum to execute this command, which it does by giving the name a$ to a part of its memory in which it then stores the string of characters between the pair of inverted commas. The result of the command is illustrated in Figure 2.4. Thus, this BASIC command can be read in plain English as 'store the

Figure 2.4 A string stored in the memory.



string of characters between the inverted commas in a part of the memory and give it the name a$'. Note that spaces count as characters just as much as letters do. When this command is executed, there is no outward sign that anything has happened. To demonstrate that the command has been obeyed, we need to know how to get at the information that has just been stored. We can print out the information stored in the a$ by typing:

```
PRINT a$
```

and when ENTER is pressed, the ZX Spectrum prints on its screen

```
Nicholas and Jonathan
```

The BASIC instruction can be understood as 'print out what is stored in a$'. Now BASIC has some features which enable us to manipulate strings of characters. By using the TO instruction it is possible to pick off characters within the string. Typing:

```
PRINT a$ (1 TO 8)
```

and pressing ENTER causes the ZX Spectrum to print:

```
Nicholas
```

while

```
PRINT a$ (14 TO )
```

and pressing ENTER causes the ZX Spectrum to print:

```
Jonathan
```

The last command can be read as 'print the last eight characters of the string of characters stored in a$'.

We have seen that BASIC makes it easy to dissect character strings, and it is just as easy to build them up. To illustrate, after storing two character strings by typing:

```
LET s$ = "sky"
```

and pressing ENTER and:

```
LET t$ = "train"
```

and pressing ENTER again, the command:

```
PRINT s$ + t$
```

gives, when ENTER is pressed,

```
skytrain
```

This command means 'print the string stored in s$ followed by the string stored in t$'. It is possible to dissect the string by using the slicing technique available with ZX Spectrum BASIC. For example PRINT s$ (1 TO 2)+t$(4 TO 5) gives on pressing ENTER

```
skin
```

In similar fashion it is possible, using these two stored words to get the ZX Spectrum to print out the words 'inky', 'tray', 'stain', 'strain', 'stinky' and several others. What happens is that only those letters of the string identified by the numbers in the brackets get used. So s$ (1 TO 2) are the first and second letters of s$ and t$ (4 TO 5) gives the fourth and fifth of t$.

The ZX Spectrum as a calculator

The ZX Spectrum can be used like a calculator in its simple command mode. If it seems to be a rather expensive calculator, remember that this is only one of the ways in which it can be used.

The number keys are situated to the top of the keyboard. The symbol ★ is used for multiplication to avoid confusion with the letter X, and / is used for division because division sums must be typed out on one line.

Arithmetic calculations can be performed by commands such as:

```
PRINT 2+3+4
```

which causes the answer to be printed, when ENTER is pressed, as

```
9
```

Similarly

```
PRINT (2★3+4)/5
```

gives, when ENTER is pressed

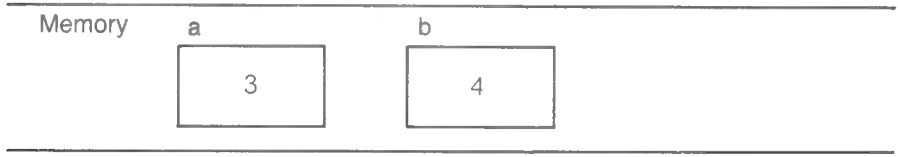
```
2
```

Numbers can be stored by

```
LET a = 3
LET b = 4
```

The result of this in memory is represented in Figure 2.5. Then

Figure 2.5 Numbers stored in the memory after a = 3: b = 4.




```
PRINT a
```

gives

```
3
```

while

```
PRINT a★b+8
```

gives

```
20
```

Printing pictures

Character strings can be made up with graphics characters, and the PRINT command can be used to generate pictures. The command

```
PRINT "<><><><>"
```

causes the pattern to be printed out. The characters in the string are alternately shifted R's and T's. This PRINT command produces output which is all on one line. Cursor control characters cannot be used in a character string, but this shortcoming can be overcome by the graphics commands explained in Chapter 4.

It can be extremely difficult to see what keys to press in order to type out character strings which contain graphics characters. For this reason a convention is adopted to indicate these characters. The conventional way to represent a graphics character is, for example, to show a shifted 4 by [^4]. A sequence of six shifted 4's is represented by [6^4]. Thus the character described would refer to a graphics character that had a black square in the bottom right hand corner of the character block. Inverse graphics characters are indicated by a double shift, i.e. [^ ^4] would give the opposite of the former character, that is, a black shape filling ¾ of the character block. Other special characters, including the cursor control characters, are represented as shown in Figure 2.6.

Figure 2.6 Symbols for graphics and control

Key	Convention
Clear screen	[^ CLS]
Graphics shifted '3'	[^3]
Graphics shifted 'e'	[^e]
Graphics shifted '3' with caps shift	[^ ^3]
Cursor down	[CD]
Cursor up	[CU]
Cursor left	[CL]
Cursor right	[CR]
Space	[SPC]
Graphics shifted space	[^ SPC]
Delete	[DEL]

To indicate more than one of the above, a number may be placed inside the brackets, e.g. [4 CL].

One method of drawing more complex graphics characters is to combine graphic strings with the AT command. If we remember that the screen is divided into 22 lines and 32 columns a character may be placed in this 'grid' by the PRINT AT line, column; 'string' command. Like most string commands they can be run together.

It is useful to use the CLS command in between the following examples as the screen will get rather cluttered otherwise!

To draw a diagonal row of four stars, for example, the command

PRINT AT 0,0;"★"; AT 1,1;"★"; AT 2,2;"★"; AT 3,3;"★".

Note that the line and column numbers start at 0,0 not, 1,1. Keying ENTER will make the pattern appear. Now key CLS and then ENTER to clear the screen.

The command to generate an open square in the middle of the screen would be:

PRINT AT 10,15; "[^ ^4] [^3] [^7]"; AT 11,15; "[^ ^5] [SPC] [^5]"; AT 12,15; "[^ ^1] [^ ^3] [^ ^2]"

Key ENTER. Now key CLS.

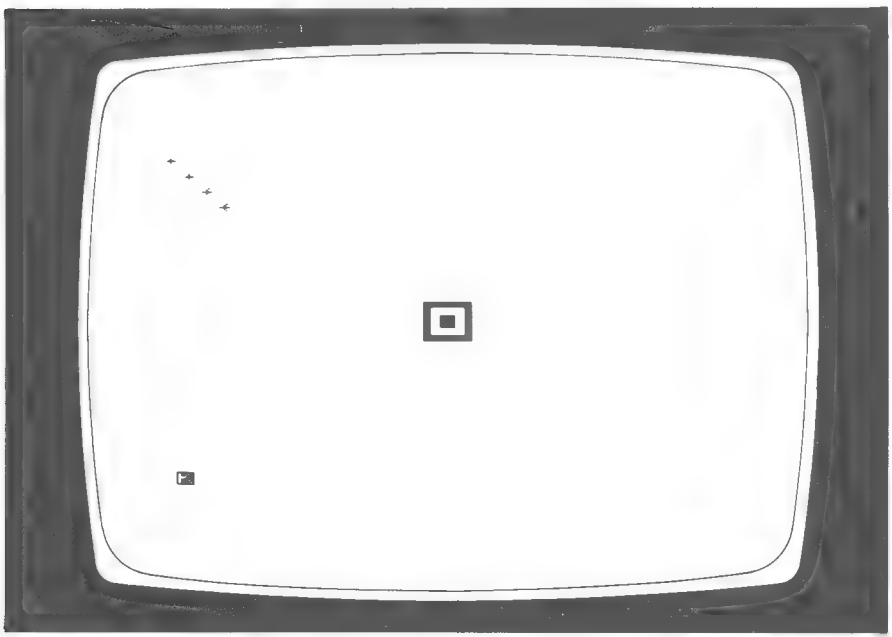
(See Figure 2.7.)

The 'Space Invader' character described earlier can be generated by

PRINT AT 10,15; "[^ ^6] [^ ^8] [^6]"; AT 11,15; "[^ ^6] [^3] [^6]";

and keying ENTER.

Figure 2.7 Two PRINT commands and their results.



Summary

A good way to become familiar with the ZX Spectrum's keyboard is to run a games program which requires responses to be made from the keyboard. A program can be loaded into the ZX Spectrum's memory from either a cassette or a disk by following the straightforward procedure initiated by giving the LOAD command. Once a program is loaded it can be run. So, when starting to use the ZX Spectrum, it is a good idea to go out and buy a few games programs and then to load and run them.

Most of the keys on the ZX Spectrum's keyboard cause the corresponding symbol to appear on the screen when they are pressed, as you would expect. However, there are some keys which are used to obtain a particular effect, such as clearing the screen or moving the flashing cursor. A little experimentation with the keyboard soon gives an appreciation of the functions of most of the keys.

Commands can be given directly to the ZX Spectrum in its own BASIC language. With the aid of just a small repertoire of commands, it is possible to instruct the ZX Spectrum to perform such diverse activities as storing and manipulating words, storing and manipulating numbers, and drawing pictures.

Self-test questions

1. What is the command which starts the procedure for loading a program into the ZX Spectrum from a cassette?
2. What commands will cause the ZX Spectrum to store the words 'lead' and 'pose'? Using these stored words only, give commands to produce the words

   'plead', 'posed', 'please', 'lose', 'addle'

   and as many others as you can find.
3. Give commands to cause the ZX Spectrum to store the numbers 4 and 5. Give commands involving these stored numbers only, and operations on them, which produce the results 16, 24 and 36.
4. Devise PRINT commands to produce the following pictures:
   (a) a right-angled triangle,
   (b) an octagon, and
   (c) a diamond mesh.

Chapter 3
Introduction to programming

Writing and running a simple program

A program is a sequence of instructions for the ZX Spectrum to obey. The language in which the instructions must be written in order that the ZX Spectrum may respond to them is called BASIC, and a few examples of individual BASIC instructions have already been introduced in the previous chapter. BASIC is a simple programming language that was devised at Dartmouth College in the USA and which first came into use in the early 1960s. It was intended to be easy to learn and to teach, and its overwhelming popularity as a language for microcomputers stems from the fact that it *is* easy to learn.

The ZX Spectrum first deals with a program by storing it, and then it can execute the program when commanded to do so. When a program is stored in the ZX Spectrum it can be executed as many times as desired, or it can be modified prior to running it again. If a BASIC instruction is preceded by a number, the ZX Spectrum treats it as an instruction belonging to a BASIC program and stores both the number and the instruction. The instruction is *not* executed at this time, but is only stored. The ZX Spectrum uses the numbers attached to the instructions to order the instructions in the program which they make up. The instruction with the lowest number is the first in the program and so on. Thus, a ZX Spectrum program consists of a set of instructions ordered according to their associated numbers.

Now let us write a short program to store the three words 'the', 'dog' and 'show', and to use these stored words to write out the phrases 'the dog show', 'show the dog' and 'dog the show'. Before starting it is a good idea to type:

NEW ENTER

because this clears any program previously stored in the ZX Spectrum. Then type in this program exactly as shown, pressing the ENTER key at the end of each line to cause it to be stored in the ZX Spectrum.

```
10 LET a$ = "the [SPC]"
20 LET b$ = "dog [SPC]"
30 LET c$ = "show [SPC]"
40 CLS
50 PRINT a$ + b$ + c$
60 PRINT c$ + a$ + b$
70 PRINT b$ + a$ + c$
```

In this program, the three words are stored at lines 10 to 30, but note that a space is included at the end of each word to act as a word separator when the phrases are printed. Line 40 causes the screen to be cleared when executed because printing a 'clear screen' character in a program has the same effect as pressing the 'clear screen' key when using the keyboard in the normal way. In fact, all the control keys behave in this way. Lines 50 to 70 cause the required phrases to be printed. Figure 3.1 shows the consequence of executing each instruction of the program. The result of executing the program is the cumulative effect produced by executing each of its instructions.

To demonstrate that the program has been stored by the ZX Spectrum, key

LIST ENTER

in response to which the ZX Spectrum always lists the program it is storing. Check the listing given on the ZX Spectrum screen against the listing above to see that they agree exactly, and, if they do, execute the program by typing: RUN, then ENTER:

```
the dog show
show the dog
dog the show
```

Remember that as the program is stored in the ZX Spectrum it can be executed or listed as often as you like. If a line of the program has been entered incorrectly, it can be corrected by listing it on the screen and using the EDIT function.

If you make a mistake when entering a program or any other command, it is quite a simple matter to change it. All lines being entered into the computer are typed in on the bottom two lines of the display. Normally these lines cannot be used for display purposes, and are not included in the display area. So although the screen can show 24 lines of 32 characters only 20 of those lines can be used. These are known as the

PAPER area of the display. The area outside of this is called the BORDER area. Whenever you press a key you enter information into the bottom of the BORDER area, and outside the PAPER area.

You will have seen already the operation of this when using a PRINT command for example. The chapter on colour graphics will deal with this in greater detail.

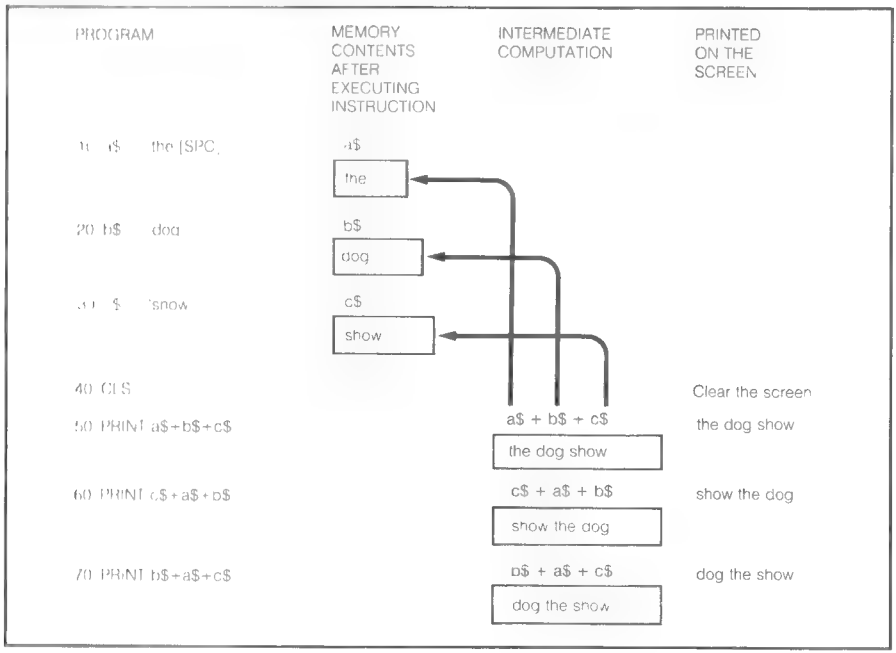
If your mistake is in the area where commands are entered then the cursor control keys will cause the cursor to move in this area. For example type in:

PRINT "FFred" (yes with the two F's)

This is obviously wrong. Use the CAPS shift key with the ← key on the 5 to move the cursor to the left. When the cursor is just to the left of the first F press the DELETE key (0) and the redundant F will disappear. Now type:

Hello [SPC]

Figure 3.1 The results of running a program.



and push ENTER. The screen should now display:

```
Hello Fred
```

So it is possible to insert text as well as deleting it. The fact that all keys have an automatic repeat function makes this very easy.

Things are slightly different though if a program has been entered. The program can be listed using either the LIST command or by pressing the ENTER key. If there is a mistake in one line, for example if line 30 had been entered as:

```
30 LET c$ = "shows [SPC]"
```

then the redundant 's' needs to be cleared. As you have been entering programs you may have noticed a forward facing arrow '>' between the line number and the text that you have just entered. This is the line cursor and indicates the last line entered. Pressing the EDIT key (CAPS shift 1) will bring this line down to the editing area. It is possible to move this editing cursor around. LIST the last program and then use the cursor control keys ↑ and ↓ to move the editing cursor up or down to the line you require. Pressing EDIT will now bring that line down for editing.

Program lines can be easily replaced. Typing:

```
30 LET c$ = "house [SPC]"
```

and pressing ENTER causes this line to replace the previous line 30, as listing will confirm. To delete a line, all that is necessary is to type the number of the line to be removed followed by ENTER. Try typing:

```
60 ENTER
```

and then list the program to see the effect it has had. When you have finished experimenting with the program, key

```
NEW ENTER
```

to delete it. After typing this, the LIST command evokes no response.

Some more BASIC instructions

In this section we meet a few more BASIC instructions. They are incorporated in short programs to illustrate their usefulness. The fact that the ENTER key has to be pressed after a command or at the end of an instruction to cause the ZX Spectrum to take the appropriate action will not be mentioned explicitly any more. As a last reminder, if you have typed something out and nothing appears to be happening as you sit and wait, it may well be that you have not pressed ENTER.

Input

Suppose that we should like to modify the program given in the first section of this chapter so that it accepts any three words we might care to give it when we run the program, and then prints out the first one followed by the second and the third, then the third followed by the first and second, and finally the second followed by the first and third. The new instruction that we need to make a program accept an input is INPUT. When an INPUT instruction is executed it causes a question mark to be printed on the screen to indicate that a response is required, and then it makes the ZX Spectrum wait until the user types a response which it then accepts. Thus, the instruction:

```
10 INPUT a$
```

produces the question mark and then if the user types:

```
the
```

the word 'the' is accepted and stored in a$. So, in this case, the effect is the same as that of the instruction:

```
10 LET a$ = "the"
```

However, the latter always makes the same assignments to the variable a$ whereas the former can assign whatever the user types.

The required program is:

```
10 CLS
20 INPUT a$
30 INPUT b$
40 INPUT c$
50 PRINT a$ + b$ + c$
60 PRINT c$ + a$ + b$
70 PRINT b$ + a$ + c$
```

When this program is executed it could result in a dialogue such as the following:

```
? the [SPC]
? kit [SPC]
? bag [SPC]
the kit bag
bag the kit
kit the bag
```

Decisions

The ZX Spectrum can be programmed to make decisions, and this ability can be used to produce some very interesting programs. The instruction which permits decision making is the IF–THEN instruction. It has the form:

IF condition THEN instruction

where in the condition part variables and/or values can be compared, typically to see if they are the same or if they differ, and the instruction part is a BASIC instruction, for example an assignment or a PRINT instruction. When executed, the condition part is tested and if the condition holds the instruction part is executed: if the condition does not hold the instruction part is not executed. An example of this kind of instruction is:

```
IF n$ = "password" THEN PRINT "accepted"
```

When this is executed, the ZX Spectrum tests if the most recent assignment to n$ is 'password': if it is, 'accepted' is printed out, otherwise nothing is done. A second example is:

```
IF n$ <>"password" THEN PRINT "rejected"
```

In this example the pair of symbols <> mean 'not equal to'. Thus when this instruction is executed 'rejected' is printed only if the most recent assignment to n$ is not 'password'.

Now consider a short program to create a sum, display it, accept an answer to it and decide if the answer is correct or not prior to printing an appropriate message. This requirement is also shown in Figure 3.2 which is an example of a *flow chart*. The program could be:

```
10 LET a = 2
20 LET b = 3
30 PRINT "what is [SPC]"; a; "+"; b; "?"
40 INPUT c
50 IF c = a + b THEN PRINT "Good. That is correct"
60 IF c<>a + b THEN PRINT "No. The answer is [SPC]"; a + b
```

Using a semicolon to separate the items in a PRINT instruction gives a different spacing than when they are printed out using a comma. Using a semicolon causes the characters inside the statement to be printed one after the other. A comma causes the items to be printed out in two columns, the second column starting in the middle of the screen.

In listing programs so far each separate statement has been given a separate line number. It is possible to join many different statements on

one line separating them with a colon. For example, the first two lines of the program above could have been written as

```
10 LET a = 2: LET b = 3
```

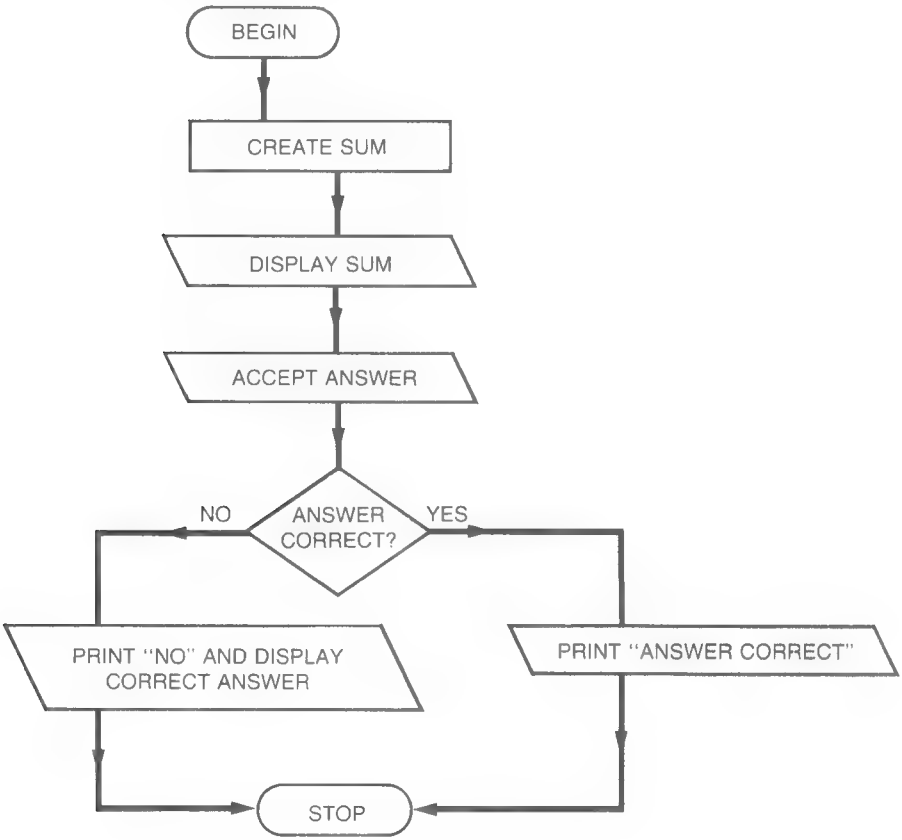
This clearly makes programs shorter, and in some cases can make them run faster.

The program can be adapted to give the user more than one attempt to find the answer, in the way illustrated by Figure 3.3 overleaf by using the GOTO instruction. The instruction

```
GOTO 30
```

means go to line 30 and execute that line next. A program that expects the

Figure 3.2 Flow chart for simple maths drill program.



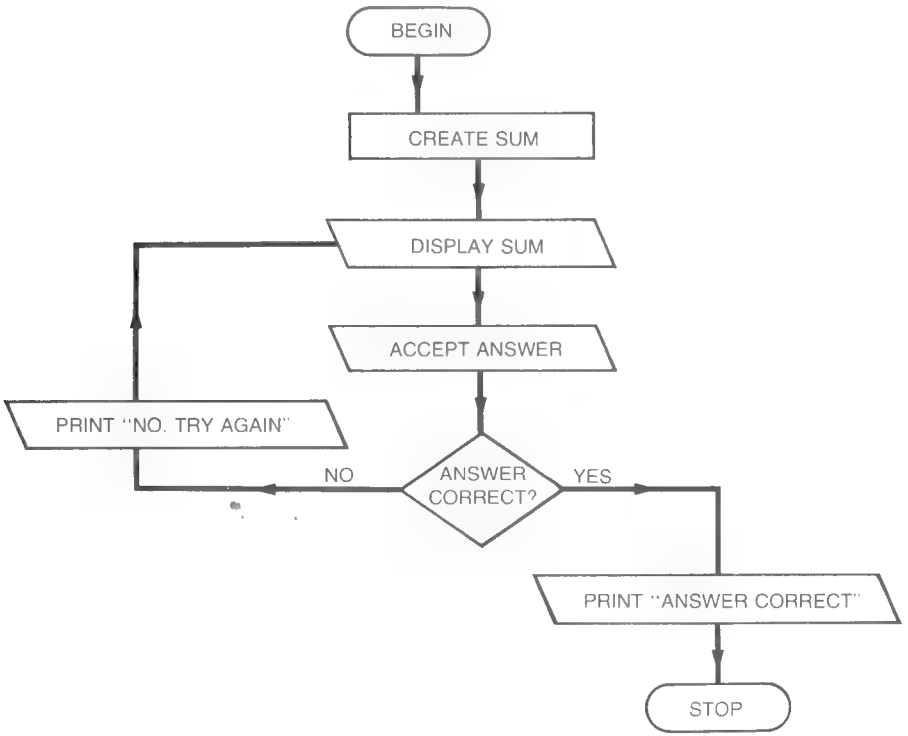
user to attempt to answer until the correct answer is given, is:

```
10 LET a = 2
20 LET b = 3
30 PRINT "What is [SPC]"; a; "+"; b; "?"
40 INPUT c
50 IF c = a + b THEN GOTO 80
60 PRINT "Sorry, wrong answer. Try again"
70 GOTO 30
80 PRINT "Good, that is correct"
```

A typical dialogue produced by this program is:

```
What is 2+3?
? 6
Sorry, wrong answer. Try again
```

Figure 3.3 Flow chart for improved maths drill program.



```
What is 2+3?
? 4
Sorry, wrong answer. Try again
What is 2+3?
? 5
Good, that is correct
```

Repetition

The previous program has shown that the ZX Spectrum can be programmed to do things repeatedly by using the GOTO instruction. The problem is posed repeatedly until the correct answer is given. BASIC has a more direct way to achieve repetition, though, and that is by the use of the FOR–NEXT instructions. To illustrate their use, the program:

```
10 FOR i = 1 TO 16
20 PRINT "Joanne"
30 NEXT i
```

causes 'Joanne' to be printed sixteen times, because all the instructions between the FOR and NEXT are repeated as many times as directed by the FOR instructions. In this case the FOR instruction directs the ZX Spectrum to do the first repetition with i=1, the next with i=2 and so on until the last repetition, with i=16, has been done. The next program illustrates that we can put as many instructions as required between the FOR and NEXT.

```
10 FOR k = 1 TO 9
20 PRINT "Repetition number [SPC]"; k
30 PRINT "Fred"
40 PRINT
50 NEXT k
```

This causes nine repetitions and produces the output:

```
Repetition number 1
Fred

Repetition number 2
Fred
```

and continues up to the ninth repetition.

We can now write a program to accept a word and spell out its letters one at a time. The program must accept a word, find its length and then repeatedly pick out and print the first, second and so on to the last letter. It works like this:

n$ (x TO x) or n$ (x) finds the xth letter of n$.

We now have all the equipment we need to write the program:

```
10 PRINT "Enter a word, please"
20 INPUT w$
30 LET l = LEN w$
40 FOR i = 1 TO l
50 PRINT "Letter number [SPC]"; i; "[SPC] is [SPC]"; w$ (i)
60 NEXT i
```

In line 30 the instruction LEN w$ determines the number of characters in the string w$.

More programs

Let us write a program to accept any name written in the form:

James Joyce

and to produce the output

Your first name is James
Your surname is Joyce

This may seem very easy at first sight, since after INPUT n$ the first name is given by n$ (1 TO 5) and the surname by n$ (7 TO 11). Unfortunately, this will produce nonsense if the name entered is Patrick Campbell, for example. What we actually need to do is to locate the position of the space separating the two parts of the name. Then, assuming that the name has been entered correctly, everything to the left of the space is the first name and everything to the right is the surname. If the name is not entered in the way we expect, strange results can still be printed, so it is sensible to request that the name be entered in a standard fashion, and then to check the entered name, rejecting it if it does not conform. This reasoning leads to the following program:

```
10 CLS
20 PRINT "Enter your name, please. Type"
30 PRINT "your first name, then one space"
40 PRINT "then your second name"
50 INPUT n$
60 LET l = LEN n$
65 LET c = 0
70 FOR i = 1 TO l
```

```
 80 IF n$ (i) = "[SPC]" THEN LET c = c + 1
 90 NEXT i
100 IF c<>1 THEN PRINT "Please enter your name as requested"
110 IF c<>1 THEN GOTO 20
120 FOR j = 1 TO l
130 IF n$ (j) = "[SPC]" THEN LET b = j
140 NEXT j
150 PRINT "Your first name is [SPC]"; n$ (1 TO b)
160 PRINT "Your surname is [SPC]"; n$ (b TO l)
```

In this program the screen is first cleared, lines 20 to 40 cause the instructions for using the program to be printed on the screen, and then line 50 accepts a name and stores it in n$. Line 60 stores the number of characters in the name in l. Line 65 sets to zero c, which will be used to count the number of spaces in the name. Lines 70 to 90 scan each character of the name, counting the number of spaces. At the end of the repetitions the number of spaces in the name is held in c. If there is not exactly one space in the name, then lines 100 and 110 indicate that the entry is not satisfactory and cause a return to line 20 to permit the name to be entered again. Lines 120 to 140 locate the position of the space, storing it in b, so that line 150 can print all the characters to the left of the space as the first name and line 160 can print all those to the right as the surname.

Our next program produces a rather fascinating mobile display of a worm-like object which moves backwards and forwards across the screen! The program starts by clearing the screen. The worm shape is stored in a$ at line 30; it is preceded by 25 spaces and followed by one space. Lines 50 to 70 cause it to move from left to right along the fourth line by repeatedly printing increasingly longer strings taken from the right of the string, that is, by printing the worm preceded by more and more spaces. Lines 80 to 100 move it in the other direction. In a FOR command such as the one in line 40 no step is mentioned, so i increases from 4 to the value of l−1 in steps of one. However, at line 70 a step is mentioned so the value of j for the first repetition is the value of −1: this value is changed by −1 until the value 4 is reached for the last repetition. If the space at the end of the string in line 20 is removed, the shape will leave a trail when it moves from right to left which is eaten up again during movement in the opposite direction. The mobile display program is:

```
 10 CLS
 20 LET a$ = "[25SPC]★★★[SPC]"
 30 LET l = LEN a$
 40 FOR i = 4 TO (l−1)
 50 PRINT AT 3,0; a$((l−i) TO l)
```

```
 60 NEXT i
 70 FOR j = (l−1) TO 4 STEP −1
 80 PRINT AT 3,3; a$ ((l−j) TO l)
 90 NEXT j
100 GOTO 40
```

Because the last line of the program always causes line 40 to be executed again, starting another pass across the screen and back by the worm, the program runs indefinitely when executed. To stop it, it is necessary to press the BREAK key.

We shall now develop a program to translate French words to their English equivalents. To do this, we shall need to store French words and the corresponding English words in such a way that they can be related. To do this, it is useful to introduce the idea of an 'array'.

In the previous section the command PRINT a$(i) was seen to print out the ith character of the string a$. By introducing the concept of the array it is possible to let many different strings of characters have the same designation. Clearly we must therefore have two variables in the statement defining the strings, one saying how many there are and the other telling the computer how much space to allocate to their storage. This is done by the use of the DIM statement.

If you type

```
DIM a$(4,5)
```

then this tells the Spectrum that there will be four strings called a$(1) to a$(4) with a maximum length of five characters. The DIM command gave the resulting array of characters its dimensions.

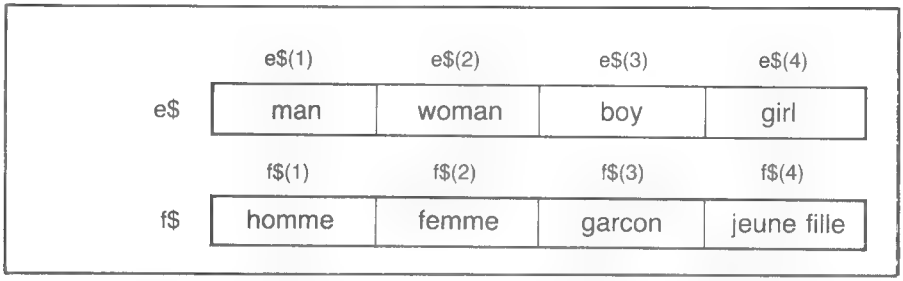
As the following program shows, arrays can be used to advantage in FOR–NEXT loops. In this program we shall use two arrays, as illustrated in Figure 3.4, with one holding English words and the other holding the equivalent French words in the same order.

The program is

```
 5 DIM w$(1,11)
10 DIM e$(4,5)
20 DIM f$(4,11)
30 LET e$(1)="man":LET f$(1)="homme"
40 LET e$(2)="woman":LET f$(2)="femme"
50 LET e$(3)="boy":LET f$(3)="garcon"
60 LET e$(4)="girl":LET f$(4)="jeune fille"
70 PRINT "Enter French word"
80 INPUT w$(1)
90 LET t=0
```

```
100 FOR i=1 TO 4
110 IF w$(l)=f$(i) THEN PRINT e$(i)
120 IF w$(l)=f$(i) THEN LET t=1
130 NEXT i
140 IF t=0 THEN PRINT w$(1); "[SPC] is not in my vocabulary"
150 GOTO 60
```

Figure 3.4 Two parallel arrays for translation program.



Clearly, as written here, the program possesses a very limited vocabulary, but this can be extended in a straightforward way. Also, it is not difficult to adapt the program so that it translates from English to French.

All the programs presented in this section can be used as vehicles for experimenting with programming in BASIC. They can be amended, extended and improved in many ways.

Saving programs

A program that is stored in the ZX Spectrum can be saved using a cassette unit so that it can be loaded again another day. When the ZX Spectrum is switched off everything stored in it is lost. To avoid having to type a program in every time you want to use it, and to preserve a record of it, it is necessary to copy it on to some form of permanent storage.

Saving a program on a cassette

To save a program stored in the ZX Spectrum on cassette, ensure that the cassette unit is attached to the ZX Spectrum and put a cassette in it. Completely rewind the tape and then wind it forward a little to avoid trying to record on the tape leader. If you are using a tape recorder with a counter you may wish to have more than one program on a cassette, in which case wind the tape so that no wanted programs are recorded over.

When the tape is in the right position decide on a name for your program. For example let us call the previous program 'translator'. It is a good idea to have only the MIC lead in. Some tape recorders do strange things with their EAR sockets and some of the signals from this may upset the computer if they get through. Set the record level about half way to its maximum and key the SAVE key followed by 'translator', making sure the quotes are included. A message will be displayed saying 'Start tape, and press any key'. The screen will then give a display similar to that when loading a program, i.e.:

5 seconds of red and pale blue stripes in the border area
A short burst of blue and yellow stripes
1 second of nothing
2 seconds of red and pale blue stripes
another burst of yellow and blue stripes

and then the message that it has loaded OK. However do not take this as being correct. This message only means that the sequence was completed successfully. It is advisable to VERIFY the program. This is done by rewinding the tape to where the recording started and keying VERIFY (E mode shifted R). Pressing the PLAY key on the recorder, and making sure that the EAR sockets are now connected, will compare the program on the tape with that in the computer. The 'OK' message means that the recording was perfect. If this message does not appear then the best thing to do is follow the checklist printed on page 23 of the Sinclair Introductory Booklet. Any number of things could have gone wrong, from the wrong recording levels to the plugs not being in correctly.

Using the printer

The addition of the ZX printer to the ZX Spectrum can enhance it considerably. Initially, the main uses for a printer are perhaps to provide program listings and to print results in a permanent and convenient form. A program listed on paper is not only a convenient record, but can also be taken away from the ZX Spectrum and studied at leisure. If a program produces a lot of results, it is more convenient to print them out than to copy them from the screen: it is much more reliable, too.

After the initial precautions of ensuring that the printer is attached to the ZX Spectrum, switched on and loaded with paper, the program stored in the ZX Spectrum can be listed on the printer rather than the screen by giving the command: LLIST.

To illustrate how the printer can be used from a program, the simple program from the beginning of this chapter is modified so that as well as giving the same output on the screen as it did before, it also produces identical output on the printer.

```
 10 LET a$ = "the [SPC]"
 20 LET b$ = "dog [SPC]"
 30 LET c$ = "show [SPC]"
 40 CLS
 50 PRINT a$+b$+c$
 60 PRINT c$+a$+b$
 70 PRINT b$+a$+c$
 80 LPRINT a$+b$+c$
 90 LPRINT c$+a$+b$
100 LPRINT b$+a$+c$
```

One additional useful feature of the ZX Spectrum is the ability to copy the contents of the screen directly to the printer. Take the previous program, for example. The same result could have been achieved just by using the COPY command, and lines 80, 90 and 100 could be left out. If the screen still has the results of that program type COPY and see what happens. (The BREAK key will be needed to stop wastage of paper!)

The COPY command is very useful when using the ZX Spectrum graphics capability, as it is far easier to draw graphics characters to the screen than to LPRINT directly to the printer.

Summary

The ZX Spectrum can store a BASIC program which can then be executed as often as required, or which can be modified before it is run again. The ZX Spectrum's BASIC language is a simple English-like language which provides, among other things, facilities for storing and manipulating information, making decisions and for repeating an action as often as necessary. In this chapter these facilities are introduced and incorporated in simple programs to illustrate ways in which they can be used. When a program has been written, it can be saved on cassette. The way in which this is done is described.

Self-test questions

1. What is the command to start the procedure for saving the program stored in the ZX Spectrum on a cassette?

2. What are the BASIC words used for:
   (a) repetition,
   (b) making a test and acting on the result, and
   (c) giving data to a program while it is running?

3. Write short programs for the following:
   (a) to print your name 10 times;
   (b) to enter a word and decide if it has more than 7 characters. If it has more than 7 characters, indicate that a long word was entered, otherwise print that it was a short one;
   (c) to accept a word repeatedly and then print it out without either its first or last letter.

4. Explain in the way illustrated in Figure 3.1 the computations performed when the following programs are executed.

(a)
```
10 LET a$ = "ALGORITHMIC"
20 LET l = LEN a$
30 FOR i = 1 TO l
40 PRINT a$ (1 TO l)
50 NEXT i
```

(b)
```
10 LET a = 1
15 LET b = 1
20 PRINT a
25 PRINT b
30 FOR i = 1 TO 12
40 LET c = a + b
50 PRINT c
60 LET a = b
65 LET b = c
70 NEXT i
```

5. Write a program to accept a word, store it in a$ and then create in b$ the reverse of the word. This can be done by starting with a string of zero characters in b$, and then adding one character at a time from the right of a$. The program should print the reversed word and then decide if the original word is a palindrome, that is, if it reads the same forwards and backwards.

A typical dialogue from the program might be:

```
Enter a word, please.
?madam
The reverse of madam is madam
madam is palindromic.
```

Chapter 4
Graphics

Most of the programs written for the ZX Spectrum that are of lasting interest and value make good use of graphics. The interest and compulsion of games and educational programs usually lie in the attractiveness of their graphics. Business programs can be much more effective than otherwise if they present information and results in pictorial as well as numerical form. Of course, some numerical computation is necessary in any reasonably complex program, whatever its application, and the results from it can be presented in one of three ways: with numbers, in words or by pictures. While in some applications it is essential to have accurate numerical results, in many others the numerical presentation of results inevitably becomes rather dull sooner or later. To present information using words is better, but books are better for reading from than video screens. Anyway, as everybody knows, a picture is worth a thousand words, and pictorial presentations are much more natural and informative than their alternatives.

As we have seen, graphic displays are achieved on the ZX Spectrum by placing graphics characters on the screen in such a way as to make up the required image. As shown in Figures 4.1 overleaf and 4.2 overleaf, the screen has 22 rows each with 32 character positions, and there is a repertoire of some 16 graphics characters. On the face of it, it might seem that only a limited range of pictures could be generated within this format, but as many existing programs have shown, and as the programs presented in this chapter demonstrate, displays of surprising complexity and detail can be produced. Some patience and ingenuity may be required to produce them, but a little knowledge and some effort are really all that is needed to start. Many people find investigating and using the graphics facilities of the ZX Spectrum one of the most interesting aspects of programming. The inclusion of good graphic effects has certainly been a major reason for the success of many of the better programs written for the ZX Spectrum, and will also help to ensure that new programs become a source of lasting pleasure and usefulness.

Methods of typing pictures on the screen and of generating simple patterns by using the PRINT command have been introduced in previous chapters. However, the ZX Spectrum supports another method which makes it relatively easy to produce graphic displays. This method is explained in the following sections.

The screen and memory

To provide a straightforward way of producing a graphic display from a program, the ZX Spectrum screen is divided up into even more picture elements, or pixels, than the number of rows and columns. As each character can be made up of 8 × 8 elements or pixels, each graphic position on the screen corresponds to a position in a 256 by 192 grid. By placing the coordinates of this location in a PLOT command, the appropriate character automatically appears in the corresponding screen position. Thus, producing a graphic display actually becomes a 'mapping' exercise.

Figure 4.1 Chart showing position of characters and pixels.

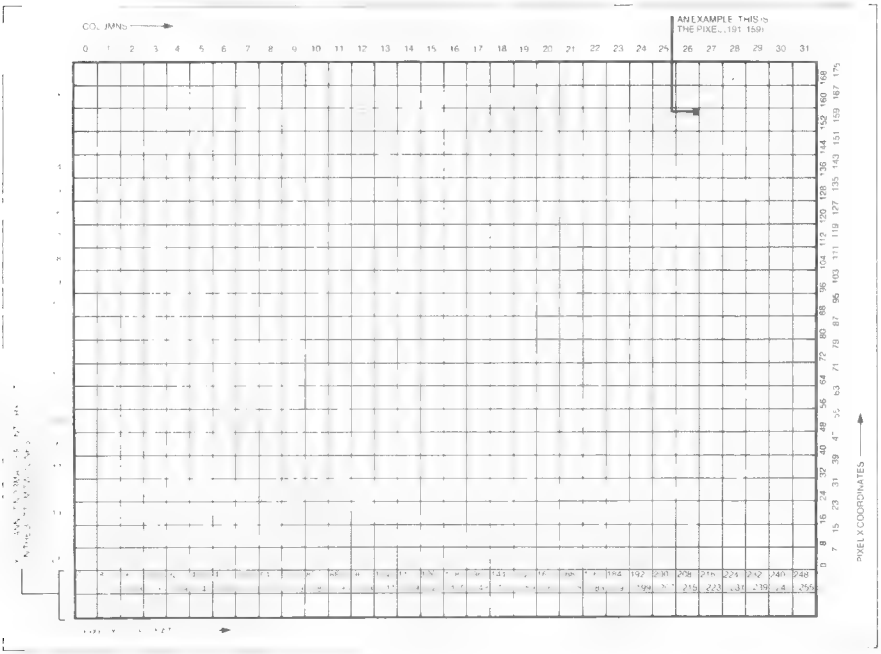
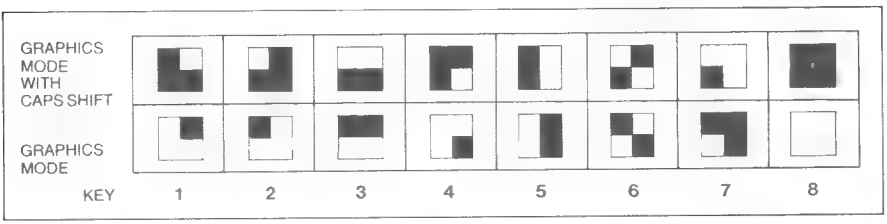


Figure 4.2 Graphics characters and their associated keys.



Putting a character on the screen

The ZX Spectrum BASIC command which enables displays to be produced on the screen is the PLOT command. The operation of this command can best be explained by an example. The command:

```
PLOT 127, 88
```

causes a dot to be placed in the middle of the screen.

Producing a drawing

A 'space invader' is an artificial image in the sense that it takes its form because of the available graphics characters rather than having a shape of its own that is modelled, or approximated, using the graphics characters. In this section a procedure for sketching a shape on the ZX Spectrum screen is described. It demonstrates that recognisable sketches can be produced, while at the same time showing that the resolution of the ZX Spectrum's display causes some problems regarding the accuracy of the sketch.

Suppose we want to draw the butterfly shown in Figure 4.3a overleaf on the ZX Spectrum screen. To do this, draw a square grid over it, as shown in Figure 4.3b, and then for each square of the grid in turn find the graphics character or pixel most closely approximating it. The result of this is shown in Figure 4.3c and the outline of the butterfly as it will be drawn on the ZX Spectrum screen is shown in Figure 4.3d. Finally, the positions for the graphics symbols are determined and a program to generate the picture is written. The simplest method of creating the 'butterfly' image would be to use the PRINT AT command:

```
PRINT AT 10, 19; "[2^ ^3]"; AT 11, 10; "[2^ ^8] [2^ ^3] [3SPC]
[^ ^3] [2^ ^8] [^ ^4]"; AT 12, 10; "[^7] [3^ ^8] [^ ^3] [^ ^8] [^ ^3]
[3^ ^8] [^ ^5]"; AT 13, 11; "[9^ ^8]"; AT 14, 11; "[^3] [6^ ^8]
[^ ^1]"; AT 15, 11; "[^5] [2^ ^8] [^ ^4] [^ ^8] [^7] [2^ ^8] [^ ^5]";
AT 16, 11; "[^1] [2^ ^8] [^2] [SPC] [^1] [^ ^8] [^ ^4]"
```

This is, however, rather tedious, especially with more complicated programs. A more interesting method is to put the coordinates of the appropriate pixels into a string and then, using the string slicing techniques considered previously, let a program do the hard work. Try this as an exercise.

It is possible to write directly to the memory locations reserved by the ZX Spectrum for storing the screen image. Each character position occupies a direct position in the memory, and the codes indicating the shape of that character can be entered from the keyboard. Unfortunately

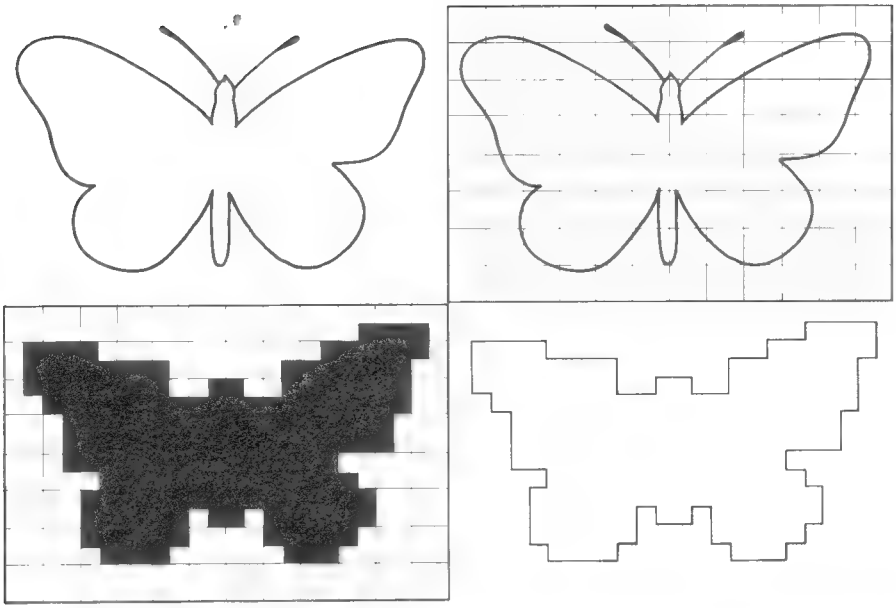


Figure 4.3 (a) Butterfly. (b) Butterfly with grid. (c) Butterfly composed of graphics characters. (d) Outline of image plotted on screen.

this is not as simple in the ZX Spectrum as with other systems, and it involves writing in the computer's machine code. For those readers interested in this aspect of programming, various books are recommended in the bibliography.

The display produced by this program is shown in Figure 4.4. The problems of resolution can be tackled in a number of ways. The simplest is to stand further away from the ZX Spectrum's screen, letting your eye and brain integrate and resolve the image as its fine detail becomes less clear! More active measures include using a smaller grid to cover the image, which gives more squares and more data for the drawing program. The positioning of the grid is important; the details which are vital from a recognition aspect should be captured as accurately as possible. Finally, a little artistic licence in the design of the displayed image may also help considerably.

The ZX Spectrum does have a number of tricks up its sleeve to make life fairly simple. For example, it is possible to draw a straight line from one place to another using the DRAW command. The Spectrum assumes that the place where it last drew a line to has the coordinates 0,0. So the left hand bottom corner starts with coordinates 0,0 after the screen has been cleared. The DRAW command has the form:

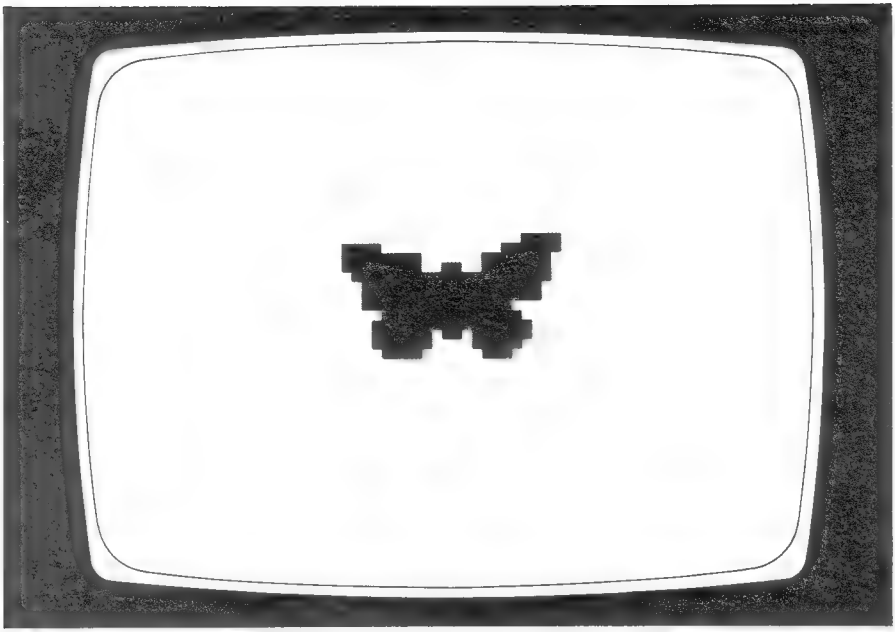


Figure 4.4 Butterfly as displayed on the screen.

```
DRAW  x,y
```

where the x part tells the Spectrum how many pixels to the right to go and the y part tells it how many pixels to go up. So type:

```
DRAW  100,100
```

and you should see a line going diagonally across the screen from the bottom left hand corner up towards the top right hand corner. Now type:

```
DRAW  0,50
```

and this will make the line go up 50 pixels.

If you want the line to go down then you have to give it a negative value of x. So now key:

```
DRAW  −50,−20
```

and the line will now come down to the left. Play around with the draw command until you are happy. Note that the Spectrum will not let you go outside its screen area of 256 by 176 pixels, and that because they start from the value 0,0 the highest value of x and y is that of the top right hand corner, i.e. 255,175. A command:

```
DRAW  255, 175
```

after a CLS command should draw a line from bottom to top.

Another useful command is CIRCLE. Like DRAW this places a circle on the screen at a point determined by the values of the x and y coordinates. But being a circle we need to tell the Spectrum its radius. So the CIRCLE command has the form:

```
CIRCLE  x,y,r
```

where r is the radius in pixels. CIRCLE does not take into account where the last pixel was plotted and assumes that x and y are relative to the bottom left hand corner. Type:

```
CIRCLE  100,100,50
```

and the Spectrum will draw a circle in the middle of the screen. Play around with this command and see if you can make some pretty patterns. DRAW and CIRCLE can be put into BASIC programs. The next program draws a series of circles up the screen:

```
10 FOR r = 0 TO 50 STEP 5
20 CIRCLE 4★r,2★r,r
30 NEXT r
```

It is also possible to get a straight line to turn through a curve. A third argument to the DRAW command tells the Spectrum how many degrees the line should turn through. Unfortunately the computer does not like degrees but uses a measure of angle called the radian. One radian is about 57°. A complete circle has 2★pi radians. So to get a line to turn through half a circle, or to draw a semicircle type:

```
PLOT 127,88: DRAW 50,50,PI
```

This command starts the line at the centre of the screen with the PLOT command and then tells the computer to draw a semicircle from 127,88 to 177,138, i.e. a radius of 25.

Using these commands it is quite simple to get a smiling face on the screen:

```
CIRCLE 127,88,50: CIRCLE 105,100,5: CIRCLE 149,100,5: CIRCLE
127,88,10: PLOT 100,76: DRAW 54,0,2
```

Screen patterns

The screen can be filled with a particular symbol by the program:

```
10 FOR y = 0 TO 21
20 FOR x = 0 TO 31
30 PRINT AT y, x; "★"
40 NEXT x
50 NEXT y
```

When the character at each screen position is generated in some systematic way, patterns that are both informative and aesthetically pleasing can be produced.

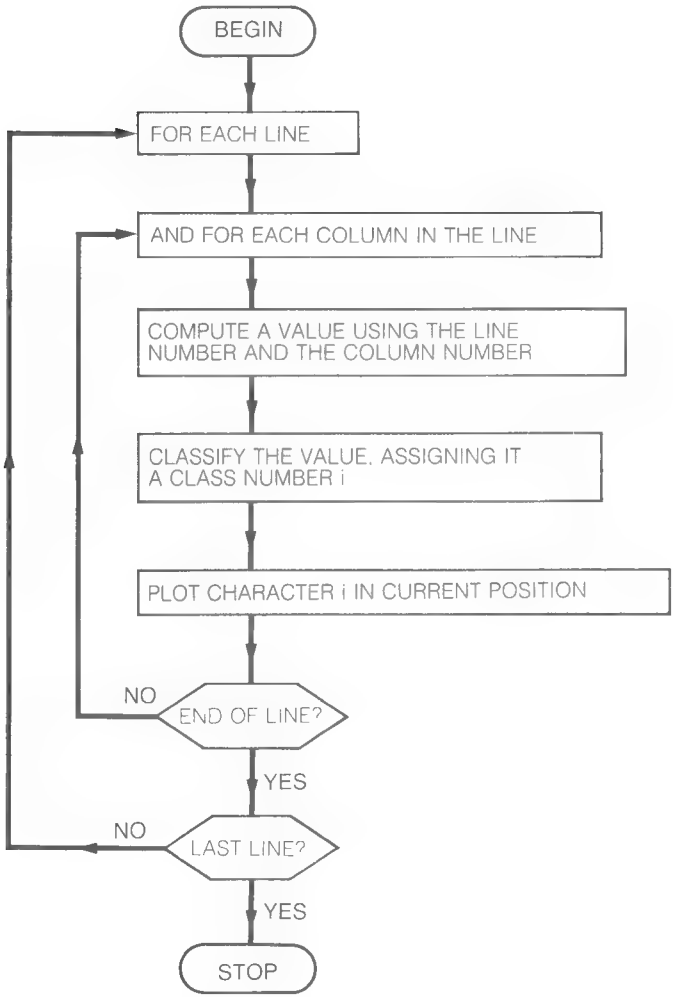
A general scheme which can be used to give a wide variety of interesting patterns involves the three stages of computation, classification and representation. A value is computed for each position on the screen using its row and column number. The value is then classified by assigning it to one of a number of classes. Each class is represented by a particular character. In this way a character can be obtained for, and plotted in, each screen position. This process is, essentially, that used to make a coloured contour map where the height of each point is measured (computed), classified into the appropriate height interval and then represented on the map by the colour assigned to that interval. A general program scheme for generating screen patterns in this way is given in Figure 4.5 overleaf. The program scheme can be refined to give a program such as the following:

```
 10 LET a$ = "I+$:"
 20 FOR I = 0 TO 21
 30 FOR c = 0 TO 31
 40 LET h = c★c + I★I
 50 IF h<80 THEN LET i = 1
 60 IF h>80 THEN LET i = 2
 70 IF h>400 THEN LET i = 3
 80 IF h>800 THEN LET i = 4
 90 PRINT AT I,c;a$(i)
100 NEXT c
110 NEXT I
120 GOTO 120
```

In this program, the codes for the plotting symbols are read into the string a$ in line 10. Lines 20 to 40 describe the computing, classification and plotting for each screen position. The computation gives a value for h in line 40, the classification is done at lines 50 to 80, and the plotting at line 90. To illustrate, for the screen position in line 10 and column 10 the value

200 is computed, this produces a value of 2 for i, and so the second plotting symbol is placed in this position. The screen display is shown in Figure 4.6.

Figure 4.5 Program scheme for screen patterns.



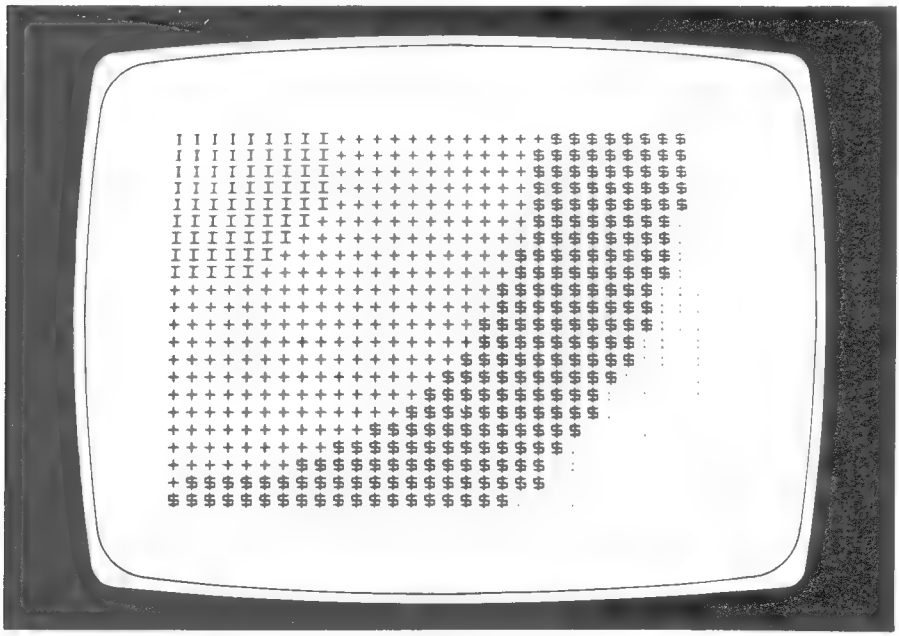
A second program following the same program scheme is:

```
10 LET a$ = "£?/★=+− ↑ %"
20 FOR l = 0 TO 21
30 FOR c = 0 TO 31
40 LET h = (l★c) ↑ (1/3)
50 LET i = INT(h)+1
60 PRINT AT l,c;a$(i)
70 NEXT c
80 NEXT l
90 GOTO 90
```

Here, there are nine intervals and plotting symbols. The computation gives l ★ c raised to the power 1/3, i.e. the cube root of l ★ c. The value is classified at line 50 by finding its integer part. As an example, take the position in row 10 and column 12: this gives l ★ c = 120, the cube root of 120 is 4.932 and its integer part is 4. So the fourth plotting symbol is placed in this position.

A wide range of patterns can be produced by using this method. In general, a distinct pattern results from each choice of computation, classification and set of plotting symbols chosen to represent the classes. Classification can be achieved in many ways other than dividing a range of

Figure 4.6 Screen pattern.



values into intervals, for example the number in the first place after the decimal point in the computed value can be used to give the class number. The selection of the plotting characters is vital to presenting the patterns effectively. The characters chosen for the last two programs are intended to accentuate the transition from one class to another, but other characters may prove more effective or attractive.

Colour graphics

One of the most exciting features of the ZX Spectrum is its ability to handle colour. Each 8×8 block of pixels that make up each character can be defined in terms of two colours. One is called the PAPER colour and the other the INK colour. In normal circumstances PAPER is white and INK is black. It is very easy to change this. Type:

```
BORDER 2
```

and the border should change to red. The colour is indicated above the appropriate number key. There are eight colours ranging from black to white as shown on the keys.

Now type:

```
PAPER 2
```

and key ENTER twice. (You need to press it twice as the colour of the PAPER and INK only change when the screen needs to be 'reformatted'). Now type:

```
INK 1
```

and ENTER twice. Now type:

```
PRINT "Blue letters on red background"
```

and ENTER. You should see just that . . . although if your television is not very good all you will get is something approximating to it. This is a very important point. Most modern televisions will accept the sort of television signals coming from computers. Unfortunately some do not. It is not noticeable on black and white but as soon as colour is used things start to go wrong. If you have persistent problems then you will have to get your local television shop to adjust it internally.

Try other combinations of INK and PAPER to see the effects. Obviously if INK and PAPER are the same colours you won't see anything!

Now it is possible to use the graphics characters to make some interesting pictures. We must remember that the most colours allowed in each character block is two, although all eight colours can be on the screen

at the same time. The following program illustrates this point. It draws a series of lines across the screen. As the screen fills up they get broader. When a pixel is plotted it is set to show the ink colour so the whole of that character block has its ink colour defined by the current ink colour.

```
10 BORDER 0: PAPER 0: INK 7:CLS
20 LET x = 0: LET y = 0: LET i = 1
30 LET a = INT (RND ★ 256): LET b = INT (RND ★ 176)
40 DRAW INK i; a−x, b−y
50 LET x = a: LET y = b
60 LET i = i + 1 : IF i = 8 THEN LET i = 1
70 GOTO 40
```

In this program the background is set to black and a random length line in colour 1 is drawn. Next time round another random line is drawn in colour 2, and so on until the ink colour becomes 8, which it cannot be so it is set back to 1.

Another program will draw the 'space invader' character but in yellow on a black background:

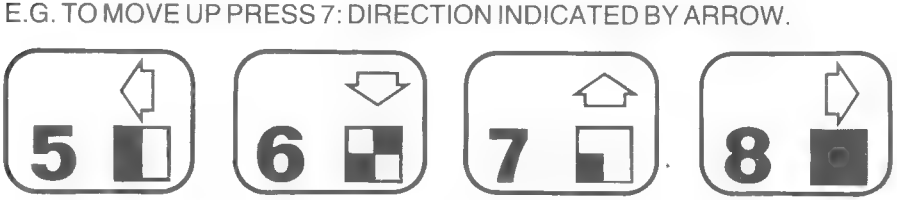
```
BORDER 0: PAPER 0: INK 6: PRINT AT 10, 15;" [^ ^6] [^ ^8] [^6]";
AT 11, 15; "[^ ^6] [^3] [^6]"
```

Movement

Once static displays can be produced, it seems natural to progress to the generation of moving displays. The program presented in this section makes it possible for the user to control the movement of a shape on the screen. Besides being fascinating in itself, this program illustrates the techniques used in many games programs.

In the better games programs written for the ZX Spectrum, a standard arrangement for movement under user control has emerged. It involves the use of numeric keys, and is illustrated in Figure 4.7. The number 7 key is used for vertical movement. The number 6 key is used to indicate

Figure 4.7 Control keys and directions.

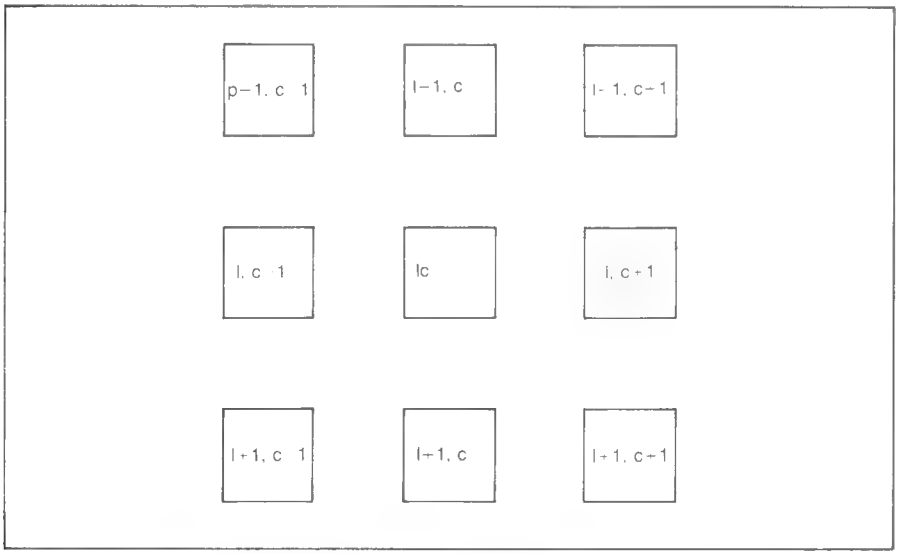


that movement is required downward, and similarly with the other two keys.

This program allows the user to draw lines on the screen. The cursor keeps moving in a certain direction until changed by the user. The program scans the keyboard to see if a control key has been pressed, and, if one has been, it directs the cursor appropriately. When the cursor is situated with reference to a screen position, lc, the changes in lc that are required to achieve movement in any direction are shown in Figure 4.8.

```
10 LET x = 10
20 LET y = 16
30 PRINT AT x, y; "★"
40 IF INKEY$ = "5" THEN LET y = y−1
50 IF INKEY$ = "6" THEN LET x = x+1
60 IF INKEY$ = "7" THEN LET x = x−1
70 IF INKEY$ = "8" THEN LET y = y+1
80 GOTO 30
```

Figure 4.8 Screen locations and directions.



There is no mechanism in this program to detect the edge of screen, and it will 'crash' if allowed to hit the side.

It is possible to make the star move without leaving characters where it has been by inserting a line deleting its previous position. This is easily done by plotting a 'space'.

Add the following lines:

```
35 LET m = x
36 LET n = y
75 PRINT AT m, n; "[SPC]"
```

The flashing display is caused by the alternating star and space being plotted.

If the program is to take into account the sides of the screen then the following lines can be added:

```
76 IF x = 22    THEN LET x = 0
77 IF x = -1    THEN LET x = 21
78 IF y = 32    THEN LET y = 0
79 IF y = -1    THEN LET y = 31
```

These lines detect whether the star is over the boundary, and if so make it loop round the 'back' of the screen and make it appear at the other end of the line.

Animation

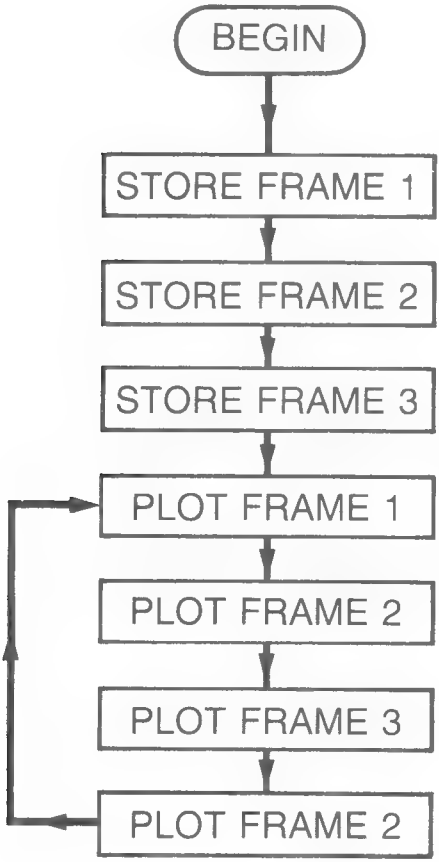
Displaying still pictures at a sufficiently high rate produces the illusion of continuous movement. All moving-picture systems, including films and television rely on this effect which depends on several human characteristics, including the persistence of vision. The program presented in this section achieves a mobile display by plotting successive static images in precisely the same way as a moving-picture is produced by showing successive still pictures. The major problem in achieving realistic mobile displays is to generate the successive static images quickly enough.

The following program produces a very simple animated display. It should be possible to get the butterfly 'flapping' its wings. However this is a very complex program when written in BASIC. The display is very 'jerky'. If programmed in machine code then the display is more effective. With the BASIC in the ZX Spectrum it is still possible to get some movement on the screen. Figure 4.9 overleaf shows a flow chart that can be used for working out a program. Let us look at making a 'space invader' cross the screen. See Figure 4.10 overleaf.

```
10 CLS
20 LET a$ = "[2^ ^8]"
30 LET b$ = "[4^ ^8]"
40 LET c$ = "[2SPC]"
50 LET d$ = "[^ ^8] [2SPC] [^ ^8]"
```

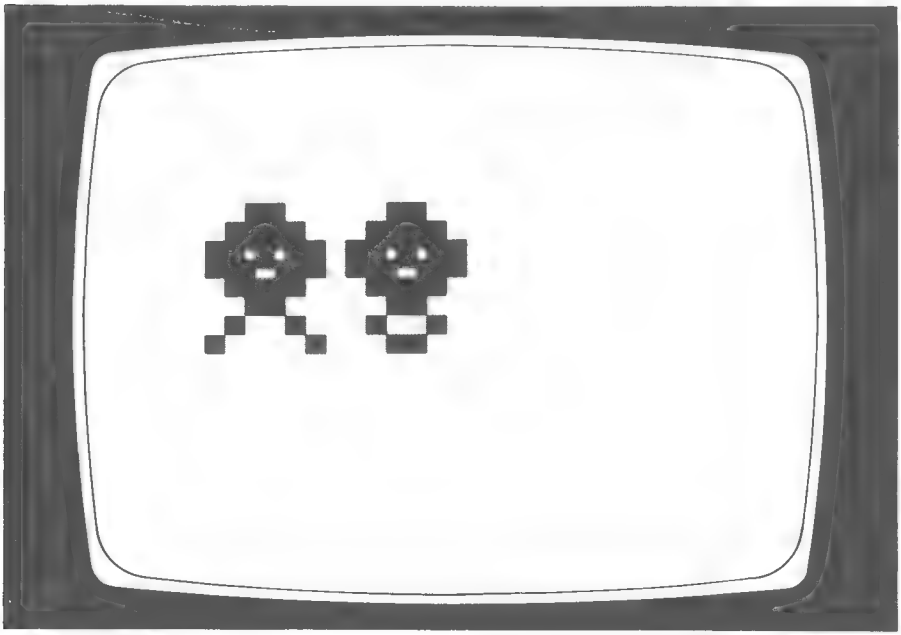
```
 60 LET e$ = "[2^ ^8] [^7] [^ ^4] [2^ ^8]"
 70 LET f$ = "[2^ ^8] [^ ^4] [^7] [2^ ^8]"
 80 LET g$ = "[^ ^8] [4SPC] [^ ^8]"
400 LET x = 7
410 LET y = 2
420 GO SUB 1000
430 LET y = y + 1
435 CLS
440 GO SUB 2000
450 LET y = y + 1
455 CLS
460 GO SUB 3000
```

Figure 4.9 Flow chart for mobile display program.



```
470 LET y = y + 1
475 CLS
480 GO SUB 2000
495 CLS
500 IF y <29 THEN GO TO 420
510 STOP
1000 PRINT AT x, y; a$; AT x + 1, y − 1; b$; AT x + 2, y − 2; e$; AT x +
     3, y − 2; f$; AT x + 4, y − 1; b$; AT x + 5, y; a$; AT x + 6, y − 1;
     d$; AT x + 7, y − 2; g$
1010 RETURN
2000 PRINT AT x, y; a$; AT x + 1, y − 1; b$; AT x + 2, y − 2; e$; AT x +
     3, y − 2; f$; AT x + 4, y − 1; b$; AT x + 5, y; a$; AT x + 6, y − 1;
     d$; AT x + 7, y − 1; d$
2010 RETURN
3000 PRINT AT x, y; a$; AT x + 1, y − 1; b$; AT x + 2, y − 2; e$; AT x +
     3, y − 2; f$; AT x + 4, y − 1; b$; AT x + 5, y; a$; AT x + 6, y − 1;
     d$; AT x + 7, y − 2; a$
3010 RETURN
```

Figure 4.10 Two 'Space Invader' frames from animation program.



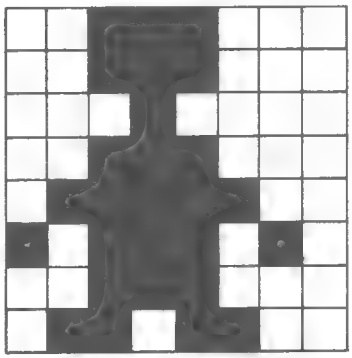
This program shows how animation can be achieved, although it does not work quickly enough to give the illusion of continuous movement. There are several methods for speeding up the plotting of the frame sequence. For instance, it is quicker to plot the changes necessary to convert one frame to the next rather than to plot entire frames all the time.

User definable graphics

The Spectrum allows the users to define their own characters. The letter keys can be given graphics characters just like the number keys. For example say that the 'a' key be defined to give the shape of a little man. Remember that the character has to be defined in a 8×8 matrix. Figure 4.11 shows how this could be done. The Spectrum uses the BIN command to enter this into the memory. The POKE command is also used. The BIN command is a method of putting binary numbers into the computer. If we let the INK colour in our character be a 1 and the PAPER colour be a 0, then the little man can be written as:

```
BIN 00111000
BIN 00111000
BIN 00010000
BIN 00111000
BIN 01111100
BIN 10111010
BIN 00101000
BIN 01101100
```

Figure 4.11 8 × 8 graphics grid — and how to translate a 'man' into BIN numbers.



0 0 1 1 1 0 0 0
0 0 1 1 1 0 0 0
0 0 0 1 0 0 0 0
0 0 1 1 1 0 0 0
0 1 1 1 1 1 0 0
1 0 1 1 1 0 1 0
0 0 1 1 1 0 0 0
0 1 1 0 1 1 0 0

This takes up eight memory locations in the computer's memory (each location can store eight ones or zeros). To get them in and also tell the computer that we want this character to be associated with the 'a' key requires the following sequence:

```
POKE USR "a" + n,BIN .....
```

where n is the row we are putting in starting with 0 and going to 7, and the numbers after BIN the 1s and 0s for that row. If this process is gone through eight times you should find that when the 'a' key is pressed in the G mode the character should appear. Now let us move it around the screen. Type:

```
10 FOR x = 1 TO 31
20 PRINT AT 10,x;"[^a]"
30 PRINT AT 10,x−1;"[SPC]"
40 NEXT x
```

Again, try experimenting with some of these ideas.

Dynamic simulation

The final program in this chapter provides a dynamic simulation of a system that experiences random growth and decay. It displays a community that grows initially from a single cell. When it reaches a certain size it decays to a lower level and then fluctuates between those two levels. The display can be taken as a simulation of the growth of a town or of a community of insects. The random element is provided by the BASIC feature RND. A figure at the bottom of the screen gives the generation of the community. This program is:

```
  5 LET g = 0
 10 DIM a (9,9)
 20 DIM b (9,9)
 30 FOR I = 2 TO 8
 40 FOR c = 2 TO 8
 50 IF RND >0.5 THEN LET a (I,c) = 1
 60 LET b (I,c) = a (I,c)
 70 NEXT c: NEXT I
 80 FOR I = 1 TO 9
 90 FOR c = 1 TO 9
 95 PRINT AT 21, 15; g
100 LET a (I,c) = b (I,c)
110 IF a (I,c) = 1 THEN PRINT AT I + 5, c + 10; "★"
```

```
120 IF a (l,c) = 0 THEN PRINT AT l + 5, c + 10; "[SPC]"
130 NEXT c: NEXT l
140 LET g = g + 1
150 FOR l = 2 TO 8
160 FOR c = 2 TO 8
170 LET x = 0
180 IF a (l - 1, c - 1) = 1 THEN LET x = x + 1
190 IF a (l - 1, c) = 1 THEN LET x = x + 1
200 IF a (l - 1, c + 1) = 1 THEN LET x = x + 1
210 IF a (l, c - 1) = 1 THEN LET x = x + 1
220 IF a (l,c) = 1 THEN LET x = x + 1
230 IF a (l,c + 1) = 1 THEN LET x = x + 1
240 IF a (l + 1, c - 1) = 1 THEN LET x = x + 1
250 IF a (l + 1, c) = 1 THEN LET x = x + 1
260 IF a (l + 1, c + 1) = 1 THEN LET x = x + 1
270 IF a (l, c) = 1 AND x <>3 AND x <>2 THEN LET b (l, c) = 0
280 IF a (l, c) = 0 AND c = 3 THEN LET b (l,c) = 1
290 NEXT c: NEXT l
300 GOTO 80
```

Sound

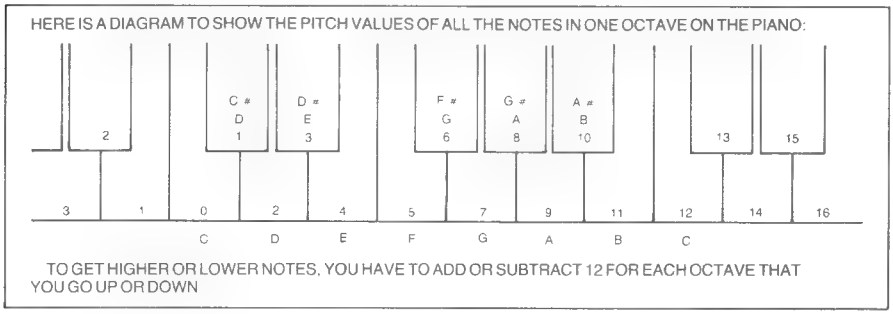
The ZX Spectrum has a very basic sound generator. (See Figure 4.12.) It gives a BEEP when the appropriate command is given. The form of the command is:

```
BEEP d,p
```

where d is the duration of the note in seconds and p is the pitch of the note relative to middle C. The first two lines of 'Three Blind Mice' can therefore be entered as:

```
BEEP 1,4: BEEP 1,2: BEEP 2,0: BEEP 1,4: BEEP 1,2: BEEP 2,0
```

Figure 4.12 How to get notes using BEEP command.



The values of d and p can be fractional, i.e. they do not have to be whole numbers. p can also be negative so that notes below middle C can be obtained. Only one note at a time is possible, and although the sound is rather quiet . . . some might say mercifully so . . . coming out of the internal speaker, the signal also appears at the EAR socket and can be plugged into a 'hifi' system if required.

One useful thing to know is that the audible feedback given when any key is pressed can be changed from the 'click' normally apparent when the system is used to enter data, etc. By POKEing location 23609 the length of the click can be altered, so that POKE 23609, 255 for instance gives a distinct sound. The length of the sound is determined by the number after the 23609.

Chapter 5
Special features of the ZX Spectrum

In this chapter, information about the ZX Spectrum and some of its special features not mentioned in previous chapters is collected together to provide a reference chapter. It is not intended to be an exhaustive collection of information about the ZX Spectrum: there are other books which provide that. It does include the features that are of interest to the person starting to use the ZX Spectrum and to anyone wanting an appreciation of its main features.

Specification of the ZX Spectrum

Manufacturer:	Sinclair Research (UK)
Microprocessor:	Z80A
Video output:	UHF TV signal giving 24 rows of 32 characters, or 256 × 192 pixels 22 rows are used for display, 2 for information
Keyboard:	40 keys, soft touch keyboard
Memory size:	16K bytes, 32K bytes expansion RAM
Languages:	When switched on, ZX BASIC is available
Graphics repertoire:	16 graphics characters, eight colours
Peripherals:	ZX printer and cassette deck. Other peripherals include
Programs:	Programs are available from a wide range of commercial sources for applications that can be classified as mainly business, educational or entertainment. Applications not falling comfortably into these categories include computer-aided design and program development

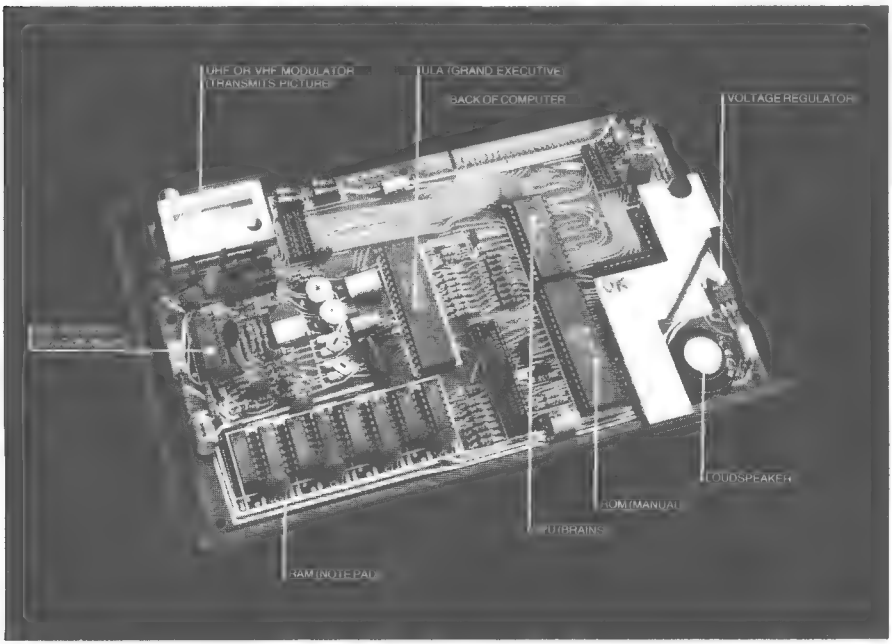
An 'exploded view' of the ZX Spectrum

Figure 5.1 shows a photographic view of the various parts that make up the ZX Spectrum. The keyboard is similar to that of a typewriter. Since these external features are familiar, most of the description in this section is devoted to the inside of the ZX Spectrum. Incidentally, while indicating the positions of components, a little more computer jargon can be introduced and explained.

Inside the ZX Spectrum are the electronics required to produce the screen display, a voltage regulator and all the electronic components that make up a microcomputer. The external power supply unit is necessary to convert the 240 volts alternating current from the mains supply to the 9 volts needed by the computer electronics. The regulator makes sure it stays at a steady value. The most convenient way to mount and interconnect the large number of components of the microcomputer is by using a printed circuit board. This has copper tracks laid down on it to connect the mounts into which the various chips are to be inserted. The layout of the printed circuit board reveals the essential structure of the microcomputer.

Mounted on the printed circuit board are the microprocessor itself, the clock and various other chips. The memory that is available to the user,

Figure 5.1 Diagram of inside of ZX Spectrum showing chips, etc.



typically for storing BASIC programs and information, is provided by 'random access memory' chips, or RAMs. The information stored in this type of memory can be accessed, and it can be replaced with other information as required. When the ZX Spectrum is switched off, all information stored in RAM is lost. Clearly, there are certain features of the ZX Spectrum which may always be required, and which it should not be possible to replace or lose when the ZX Spectrum is turned off. For example, BASIC should always be available, and the characters to be displayed on the screen may need to be generated at any time. These functions are provided by chips with information permanently stored in them. The chips are known as 'read-only memories' or ROMs. The positions of the BASIC ROM and the character generation ROM are shown in Figure 5.1.

Special features

In this section three of the more useful and interesting of the ZX Spectrum's features not so far introduced are described.

The ZX Spectrum's clock

The ZX Spectrum has an in-built clock which can be accessed from a BASIC program. The time on the clock is obtained by using the PAUSE command. On a 50Hz system as used in Europe the command PAUSE n will stop any program for n × 50 seconds. n has a maximum value of 32767, so the longest pause possible is 10 hours 55 minutes 21 seconds. Unfortunately, if the PAUSE command is used in a program then the time that the system takes to work out the program has to be added to the PAUSE n value. For example:

```
10 LET s = 0
20 PRINT AT 10, 15; s
30 LET s = s+1
40 IF s = 60 THEN LET s = 0
50 IF s = 0 THEN CLS
60 PAUSE 20
70 GOTO 20
```

This program counts seconds and returns to zero at the end of one minute. The number after PAUSE in line 60 has to be worked out by experiment. As an exercise see if the hours and minutes elapsing can be displayed.

A more accurate method of counting elapsed time is by looking at the contents of certain memory locations. It is quite complicated and involves PEEKing the contents of three locations. The PEEK command is exactly

the opposite of POKE in this respect. The instruction:

```
(65536*PEEK 23674 + 256*PEEK 23673 + PEEK 23672)/50
```

gives the number of seconds since the computer was switched on. The memory locations used are incremented by one each time the screen is scanned, i.e. every 50th of a second. This expression can be incorporated into a BASIC program if required.

Examining the contents of a location

We have seen that the contents of any location can be examined by using PEEK. The command:

```
PRINT PEEK (16416)
```

prints the data stored in location 16416, while

```
LET x = PEEK (16397)
```

assigns the data stored in location 16397 to x.

Available storage

At any time the amount of memory that is available can be found by using the command:

```
PRINT PEEK 23613 - PEEK 23653 + 256*(PEEK 23613 - PEEK 23653) -100
```

This looks at the various memory locations used to tell the microprocessor where things are. The calculation uses these parameters to give a rough idea of the number of bytes left. (The information on the screen takes up a discrete amount of space!)

It is also useful to know how much storage a program requires, as this can prevent such frustrations as, for example, trying to read a program requiring 18K of store on a 16K ZX Spectrum, or trying to extend a program which uses almost all the available memory. The command:

```
PRINT PEEK 23627 + 256*PEEK 23628 - 23755
```

gives the memory used by the program.

The user port

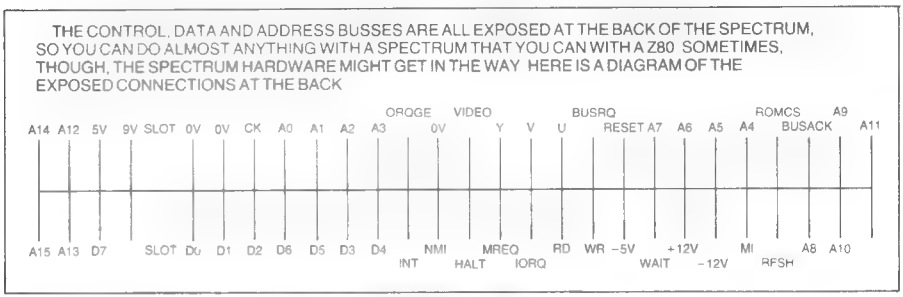
The way in which the expansion port can be used to connect the ZX Spectrum to other equipment so that it can communicate with it, or control it, is outlined in this section. A full treatment requires the use of binary numbers and their arithmetic. There is nothing especially difficult

about this, but it lies beyond the scope of this book.

The user port is essentially an extension of the computer 'bus' – or control data and address lines. There are also outlets for the power supply. Peripheral units sit on this bus and communicate with the microprocessor and the other electronics via it. The ZX printer also sits on the bus, and data to be printed also travels along it. Each peripheral recognises its own data by the address on the address lines. So it should be impossible for data going to the printer to get mixed up with data going to and from the memory.

As shown in Figure 5.2, the sockets for connecting the ZX Spectrum to other devices are provided by connections at the edge of the printed circuit board. The edge connector on the printed circuit board permits the expansion of the ZX Spectrum.

Figure 5.2 Edge connector allocation.



How the ZX Spectrum stores a program

A BASIC program is stored in the ZX Spectrum starting at the location given by the numbers stored in locations 23635 and 23636. Normally this number is 23755, although if microdrives or any communications devices are being used, this can change. It is stored as a linked list of program lines as illustrated in Figure 5.3 overleaf. Each character and BASIC word is represented by a code: a different code from that used in conjunction with POKE is employed. Each line of the BASIC program is stored character by character and BASIC word by BASIC word starting with the line number. The end of the line is indicated by the length of the text value.

To give an example to illustrate this, the short BASIC program:

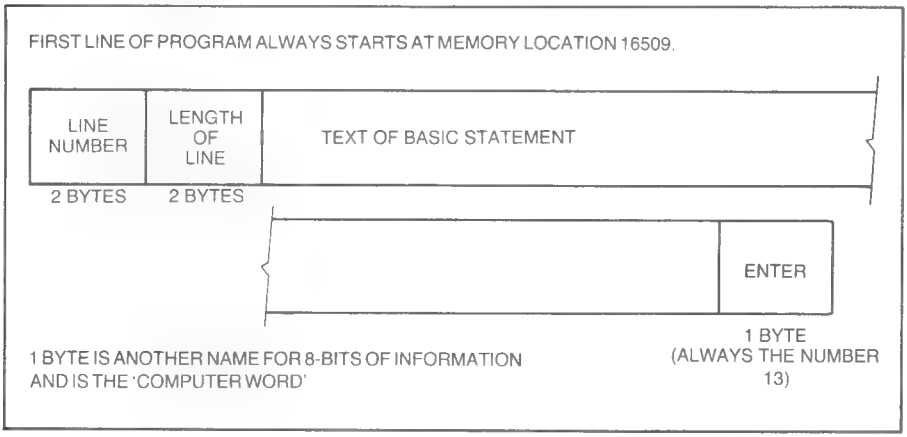
```
10 LET p = 1
20 LET q = p + 2
30 PRINT p,q
```

is stored as shown in the following table.

TABLE 5.1

Location number	Code	Character or BASIC word
23755 23756	0 10	Line number 10
23757 23758	11 0	No of bytes in line
23759	241	LET
23760	112	p
23761	61	=
23762	49	1
23763	14	number
23764 23765 23766 23767 23768	0 0 1 0 0	number 1 in computer's code
23769	13	ENTER
23770 23771	0 20	Line number 20
23772 23773	13 0	No of bytes in line
23774	241	LET
23775	113	q
23776	61	=
23777	112	p
23778	43	+
23779	50	2
23780	14	number
23781 23782 23783 23784 23785	0 0 2 0 0	2 in computer's code
23786	13	ENTER
23787 23788	0 30	line number 30
23789 23790	5 0	number of bytes
23791	245	PRINT
23792	112	P
23793	44	,
23794	113	q
23795	13	ENTER

Figure 5.3 How the ZX Spectrum stores a program.



The form of this BASIC program when stored is illustrated in Table 5.1. The codes for the characters and BASIC words can be observed by using PEEK to examine the area in which the program is stored.

Programs are stored in this way to facilitate the procedures by which the ZX Spectrum stores program lines in the correct order regardless of the order in which they are entered, and subsequently, deletes or inserts lines as required.

Special locations

The next table lists the special locations in the ZX Spectrum's memory that have been referred to in this and previous chapters. It also describes the purpose for which these locations are used.

TABLE 5.2

Location numbers	Purpose
23732	Address of last byte of the BASIC system area
23617	Cursor mode k, l, g, c or e
23621	Line number of statement currently being executed
23625	Number of current line
16384	Address of start of screen memory
23560	Shows which key has been pressed
23637	Address of next program line to be executed
23672	Counts the number of frames displayed on television
23677	x-coordinate of last point PLOTted
23678	y-coordinate of last point PLOTted
23296	Printer buffer
23562	Delay (in 50ths of a second) between repeats on a key (normally 5)

Appendix 1

Further reading

This appendix gives some books and magazines that are suitable for further reading to follow up particular topics that are mentioned, introduced or developed in this book. They are mainly for the ZX81; the programs can be 'adjusted' quite easily – see Appendix 2.

General books

1. *The Gateway guide to the ZX81 and ZX80* by Mark Charlton (Database Consultancy, 1981).
A good introductory book aimed at those wanting to get involved with programming. Lots of games programs.

2. *Hints and Tips for the ZX81* by Andrew Hewson (Hewson Consultants, 1981).
Contains many routines useful for programmers. Display explained in detail.

3. *Getting acquainted with your ZX81* by Tim Hartnell (Database Consultancy, 1981).
Many programs with some useful information scattered through the text.

4. *Understanding our ZX81* by I. Logan (Essential Software Company, 1981).
Illustrates all the attributes of the ZX81 Monitor, how it works, and how to use it in writing useful programs.

Games

5. *50 Rip-roaring games for the ZX80 and ZX81* by Jeff Weinrich (Database Consultancy 1981).
The title says it all!

6. *49 Explosive Games for the ZX81* by Tim Hartnell (Interface, 1981).
Galactic Intruders, Breakout, Draughts, Star Trek, etc., etc.

7. *30 Programs for the ZX81 . . . 1K* (The Essential Software Company, 1981).
As the title explains this book contains 30 games for the ZX81, and also some useful hints and advice on writing programs.

Programming

8. *Stretching your ZX81 or ZX80 to its limits* by Tim Hartnell and Trevor Sharples (Computer Publications, 1981).
Information on how to improve programming efficiency.

9. *Learning BASIC with your Sinclair ZX80* by Robin Norman (Newnes Technical Publications, 1981).
BASIC programming for the ZX80 with the new 8K ROM.

10. *ZX81 BASIC Book* by Robin Norman (Newnes Technical Publications, 1981).
Covers the 1K and expanded 16K versions of the system.

11. *BASIC Programming for the ZX81* by Ian Stewart and Robin Jones (Shiva Publishing, 1982).
Covers most of the features in Sinclair, BASIC, and also contains 50 games programs.

Education

12. *Educare's 50* by K. S. Goh (Educare, 1981).
Fifty programs that can be used in education, all aimed at the primary school market.

Machine Code

13. *Mastering machine code on your ZX81* by Tony Baker (Interface, 1981).
A very comprehensive look at the inside of the computer. Not for the faint-hearted or inexperienced!

Applications

14. *The Sinclair ZX81 – programming for real applications* by Randle Hurley (Macmillan, 1981).
From word processing, through home banking to education . . . a book for those wanting to go beyond playing games.

Magazines

15. *Sinclair User* (ECC Publications, 30-31 Islington Green, London N1).
A magazine aimed at the beginner.

16. *Interface* (44-46 Earl's Court Road, London W8 6EJ).
Monthly magazine of the National ZX80 and ZX81 User Club. Started off with some useful ideas. Since taking in the Acorn Atom seems to be suffering from a kind of journalistic schizophrenia!

17. *Computer and Video Games* (EMAP, 8 Herbal Hill, London EC1).
Many games programs for ZX81 users.

18. *Computing Today* (ASP, 145 Charing Cross Road, London W1).
Monthly general purpose computer magazine with lots of news, games and reviews.

Appendix 2

Differences between ZX81 and ZX Spectrum BASIC

The ZX81 and the ZX Spectrum are essentially very similar computers. A few differences do occur within the BASIC. If machine code is considered, though, then there are great differences and it is *not* easy to convert such programs. This means that some of the books in the previous appendix will be of only general use.

READ, DATA, RESTORE

The ZX Spectrum has the BASIC commands READ, DATA and RESTORE. The ZX81 does not. Many ZX81 programs that need this construction use string arrays to overcome the problem. Another method is to use a whole series of LET statements. Any programs for the ZX81 using either of these methods will run without any change.

SLOW and FAST

The ZX Spectrum does not need either of these commands that are found on the ZX81. The Spectrum can be considered to have the speed of the ZX81 in FAST mode with the screen attributes of SLOW mode. Consequently these commands can be taken out of any ZX81 programs.

SCROLL

The ZX Spectrum scrolls automatically so the ZX81 command for this effect can be taken out. The ZX Spectrum asks 'scroll?' every time the screen is filled.

UNPLOT

UNPLOT does not exist on the ZX Spectrum. The nearest equivalent is the PLOT OVER command. This only works on an individual pixel, whereas the PLOT and UNPLOT commands on the ZX81 work on a 4×4 group of pixels. Some thought is necessary before programs are converted.

Graphics in general

The ZX81 has a similar screen size to the ZX Spectrum. So the PRINT AT commands and positions are identical. The ZX81 has a larger pixel size and thus only PLOTS on a 64×44 grid. This compares to the 256×192 grid of the ZX Spectrum. The ZX Spectrum also uses a PLOT command, but this applies to the individual pixel not to the 4×4 group. Some thought must be given to the problems caused by this. For example consider the following ZX81 program:

```
10 FOR y = 12 TO 33
20 PLOT 31,y
30 NEXT y
40 FOR x = 15 TO 47
50 PLOT x,22
60 NEXT x
```

This program will plot a cross in the middle of the screen. The simplest way to get the same effect with the ZX Spectrum is to use the OVER command and the graphics symbols for the ¼ squares. The program then becomes:

```
10 FOR I = 5 TO 15
20 PRINT AT I,16:"[^ ^5]"
30 NEXT I
40 FOR c = 10 TO 23
50 OVER 1: PRINT AT 10,c;"[^ ^3]"
60 NEXT c
```

This is very much a simple example to indicate some of the problems involved.

Character set

The ZX Spectrum uses a standard ASCII character set. The ZX81 uses a non-standard set. Consequently any peripherals, like printer interfaces, that contain electronic circuitry to convert the non-standard set will not work on the ZX Spectrum.

Cassette compatibility

Finally there is absolutely no compatibility between the two cassette recording formats. No ZX81 cassette can be used on the ZX Spectrum,

and vice versa. This does mean that a lot of good software will have to be translated from the ZX81 to the ZX Spectrum. However with the interest and ingenuity of those who made the ZX81 software marketplace so exciting it is only a matter of time before this is rectified. No doubt somebody will invent a small box of tricks that can be plugged in the back and do the translating for us.

Appendix 3

Glossary

Array
A block of sequential segments of memory reserved in a program by, for example, DIM A$(BC). They are named, in this case, A$(1), A$(2) . . . A$(B) giving a set of names which can be used in the same way as ordinary variable names, but with the convenience that they include a bracketed index. C indicates the maximum length of the strings.

BASIC
The computer language available when the ZX Spectrum is switched on, and in which commands to it are expressed. BASIC actually stands for Beginner's All-purpose Symbolic Instruction Code.

Byte
Strictly, a group of binary digits but for simplicity it can be regarded as a memory location whose contents can be any one of 256 possibilities.

Chip
Literally, the chip of silicon from which an integrated circuit is fabricated, but used popularly to refer to the integrated circuit itself.

Cursor
The flashing square on the ZX Spectrum's screen which indicates the position at which the next item will be displayed.

Database
An organised collection of data from which either data or the properties of items of data can easily be retrieved.

Disk
A disk on which programs or data can be stored as magnetic patterns on the surface of the disk, and from which recorded information can be rapidly retrieved. Also known as a floppy disk.

DOS
Disk Operating System. A program to facilitate the storage of information on disk and its retrieval from the disk.

ENTER
When the ENTER key is pressed at the end of a line, that line is sent to the ZX Spectrum to be dealt with. For example, commands are then executed, and program lines are stored.

Flow chart
A diagram indicating in stylised form the steps of a computation. It is used as an aid in developing programs.

Graphics
Pictures produced by a computer.

Integrated circuit
An electronic circuit fabricated in extreme miniature form on a silicon chip typically a few millimetres square.

K
1K stands for 1 kilobyte of memory and gives the size of a memory consisting of 1024 storage locations.

Machine code
The code in which instructions must be conveyed to a microprocessor in order that it may respond to them directly.

Microprocessor
Physically, a very complex integrated circuit. Functionally, an electronic device that can be programmed and can therefore perform a variety of tasks.

Peripheral
Equipment that can be attached to the ZX Spectrum, and can be used in conjunction with the ZX Spectrum because the latter can control it. Examples are cassette units and printers.

Printed circuit board
A board on which conducting connections and sockets are mounted so that it can support and interconnect electronic components and integrated circuits.

Program
An ordered sequence of commands given to a computer so that when it obeys them it automatically performs a specified task.

RAM
Random access memory. Memory whose contents are lost when the power supply is turned off. The amount of RAM determines how much memory is available for the user to store programs and data.

ROM
Read-only memory. This is permanent memory, typically used to store information that is always required, such as that which provides BASIC.

This memory is not available to store the user's programs: it provides facilities required by the user.

Software
Software means programs, although it includes utility programs as well as the user's own programs. This contrasts with hardware, which refers to the physical equipment of a computing system.

User port
One of the connections at the rear of the ZX Spectrum, which can be used to send or receive signals under the control of the user's program.

Word processor
A system for processing textual material electronically and then printing it or, perhaps, transmitting it to a similar system. In this context, the processing is mainly editing.

Index